Department of Computer Science
George Mason University Technical Reports

4400 University Drive MS#4A5
Fairfax, VA 22030-4444 USA
http://cs.gmu.edu/ 703-993-1530

A Taxonomy of Job Scheduling on Distributed Computing Systems

Raquel Lopes
raquel@computacao.ufcg.edu.br

Daniel Menascé
menasce@cs.gmu.edu

Technical Report GMU-CS-TR-2015-13

Abstract

Hundreds of papers on job scheduling for distributed systems are published every year and it becomes increasingly difficult to classify them. Our analysis revealed that half of these papers are barely cited. This technical report presents a general taxonomy for scheduling problems and solutions in distributed systems. This taxonomy was used to classify and make publicly available the classification of 100 scheduling problems and their solutions. These 100 problems were further clustered into eight groups based on the features of the taxonomy. The proposed taxonomy will facilitate researchers to build on prior art, increase new research visibility, and minimize redundant effort.

1 Introduction

In the last decade, cluster computing emerged as the main platform for high performance, grid, and cloud computing. Together, these three different, yet very similar platforms, emerged as important sources of computing power. They all consist of distributed computers (or nodes) connected through high speed networks.

Scheduling parallel jobs in distributed systems is an NP-hard problem that has been attracting the attention of researchers for decades. Thousands of solutions have been published; however, these solutions deal with slightly different versions of a scheduling problem. Indeed, there are many knobs that may be tuned in order to clearly specify a scheduling problem of this nature. To the best of our knowledge these knobs have not been defined for general scheduling problems. In 1995, a very important research in the area indicated the need for the proper definition of scheduling problems:

> At the very minimum, we wish that all papers about job schedulers, either real or paper design, make clear their assumptions about the workload, the permissible actions allowed by the system, and the metric that is being optimized. *[1]*

Unfortunately, twenty years later the situation has not improved. So far, the many knobs needed to define a scheduling problem have been tuned on an *ad hoc* individual basis. It is time for change. While hundreds of papers on scheduling are published every year, it becomes increasingly difficult to easily identify scheduling problems and classify new scheduling solutions. To the best of our knowledge, a general taxonomy to define job scheduling problems and solutions in distributed systems does not exist. This technical report aims at shedding light on this unstructured scenario by defining a general taxonomy of scheduling problems and solutions in distributed environments.

Early seminal work aimed at defining taxonomies to classify scheduling problems and solutions exist. An important work written in 1988 defines a taxonomy for distributed job scheduling solutions [2]. Another inspiring work defines a language to define general scheduling problems [3]. In spite of the inspiring nature of these seminal propositions, a general taxonomy that takes into account the new generation of distributed systems and scheduling solutions is required. It is time to think about the properties that describe scheduling problems and solutions considering the current scenario.

More recently, some researchers became concerned again with this problem. They have defined taxonomies for specific types of distributed platforms. However, none try to cover a distributed system in general, as we argue is the most appropriate solution. The authors of [4] define a taxonomy of scheduling problems in grid computing platforms. Smanchat and Viriyapant [5] extend and complement the grid taxonomy in order to define a new taxonomy of scheduling problems in cloud computing. These taxonomies overlap in some aspects, especially those describing workload and solution, and at

the same time, they are over-fitting models, not general enough to be applied to any kind of distributed platform known today. They consider properties that represent very specific details of each resource platform. For example, the grid taxonomy [4] only considers scheduling problems that target multi-criteria decision analysis involving cost. This excludes many scheduling problems in which cost is not considered or in which the scheduling goal considers one criterion, like minimization of makespan, that is historically the most popular scheduling criteria. Some properties are highly coupled with grid environments such as the cost model flexibility, and intra and interdependence among scheduling criteria. The taxonomies of workflow scheduling techniques in the cloud assume that resources are virtual machines, which is not true for all distributed platforms, even for the cloud [1]. Some properties of the cloud taxonomy are highly coupled with traditional cloud environments, such as VM startup latency and provisioning model (on-demand, reservation or spot).

It is also important to point out that the taxonomies mentioned above fail to consider some properties that are important to clearly define scheduling problems and solutions. For instance, they do not define workload composition in a complete fashion, neither resource sharing or scaling. They also do not consider important requirements such as data locality and failure model. Finally, they do not include properties that characterize the quality of service required by the workload. We argue that these and other features must be considered.

We concluded that prior work in scheduling taxonomies is not sufficiently generic or complete enough for classifying scheduling problems and solutions in distributed platforms. They either focus on specific resources categories and not distributed resources in general. We argue that a unified taxonomy is possible and, in fact, needed, in opposition to many specific overlapping taxonomies for each different type of distributed platform. As a matter of fact, there are more common features among these platforms than specific ones. Moreover, it is increasingly common to consider hybrid infrastructures, in which in-house resources are complemented with resources from the cloud or grid platforms. A unified taxonomy would cover all these cases. Finally and most importantly, it is easier to maintain a single taxonomy over the years than to maintain many different, overlapping ones.

For these reasons, we have defined our own taxonomy to help classify existing (and future) scheduling problems and solutions. The taxonomy targets the scheduling of parallel jobs in distributed systems. The solution is clearly meaningless without the associated problem. The problem, however, can be useful alone for comparison reasons. So, we organize the taxonomy in such a way that the problem and the solution can be easily separated. We propose the use of the taxonomy to (i) instantiate different scheduling problems and (ii) classify different scheduling solutions. The proposed taxonomy is built upon existing work surveyed in Section 7.

The contributions of this technical report are four-fold:

1. A comprehensive taxonomy for classifying scheduling problems and solutions. This taxonomy allows a researcher to define what is claimed, i.e., which portion of the scheduling problem space is being addressed and to define the properties of the scheduling solution in a comprehensible fashion. This taxonomy provides a snapshot of the state-of-the-art of parallel job scheduling in distributed systems;

2. An analysis of the impact of a subset of 1050 papers related to parallel job scheduling in distributed systems from 2005 to 2015 (May, 1^{st});

3. A classification, using the taxonomy, of scheduling problems and solutions of the top-100 papers in the area, considering the number of citations per year.

4. An online scheduling archive, collaboratively constructed, in which classified scheduling problems and solutions may be found and others may be added.

We found that almost 22% of the papers related to parallel job scheduling in distributed systems are never cited. A Pareto behavior is somehow seen: 12% of the papers in the area are responsible for 64% of all citations. 40% of the papers are cited at most twice in their entire life. More than half of the papers are not cited more than four times in their entire life. This is a sad indication that we are still crawling towards a real scientific methodology. We hope that by classifying the papers using well-known taxonomies, researchers will be able to clearly indicate what kinds of problems and solutions they are claiming. As a consequence, the classification will allow new research to be built on top of the prior art and it will be easier to know the state-of-the-art regarding specific instantiations of scheduling problems.

Richard Hamming, in 1968, figured out what is the central problem of Computer Science during his Turning Award Lecture:

> Perhaps the central problem we face in all of computer science is how we are to get to the situation where we build on top of the work of others rather than redoing so much of it in a trivially different way. Science is supposed to be cumulative, not almost endless duplication of the same kind of things. *[6]*

It is our belief that building an adequate taxonomy constitutes a first step towards the direction pointed

[1] Indeed, Metal as a Service (MaaS) has recently arisen as a new cloud computing model in which virtualization is not used because the cloud user wants to deploy directly onto bare metal for optimum performance. OpenStack, for instance, a popular open source cloud platform is already considering this new model (see https://wiki.openstack.org/wiki/Ironic).

by Hamming. Without proper mechanisms to classify work we are doomed to ignore what others have done. Many other steps are still necessary. In particular, the discipline to use the taxonomies from now on and the need to maintain the taxonomies up-to-date.

An important action in this regard is to maintain an archive of scheduling problems and solutions based on the taxonomies. For that purpose, we created a web site, the DSS Archive (Distributed Systems Scheduling)[2]. We initially populated the site with the classification of 100 problems. The idea is to collaboratively increase the number of papers cataloged. The site offers two other services: (i) search for research papers that tackle a specific scheduling problem and (ii) find the distance between two scheduling problems.

The rest of this report is organized as follows. Section 2 presents a background on scheduling theory and a high-level definition of what is a scheduling problem. Section 3 introduces a taxonomy for scheduling in distributed systems that contemplates both problems and solutions. Section 4 summarizes the research method and underlying review protocol while leaving the details to Appendix A. This review protocol was used to collect 1050 papers related to scheduling parallel jobs in distributed systems. The next section presents overall statistics about these papers including popularity and number of papers published per year during the last decade. The taxonomy was used to classify one hundred scheduling problems and respective solutions. The results are summarized in Section 6; a list of these 100 problems can be found in Appendix B. Related work is discussed in Section 7. The report concludes in Section 8 with a set of recommendations for future research on scheduling in distributed systems.

2 Background on Scheduling Theory

This section provides a conceptual model of scheduling problems and solutions in distributed computer systems. Some definitions in this section are based on previous work [7, 8]. We assume the reader is generally familiar with scheduling. We do not consider in this report single-node scheduling problems, which have been thoroughly investigated in the field of operating systems.

Scheduling is the *assignment of resources to consumers in time*. In general, every instance of a scheduling problem must clearly specify three components:

- **Workload**: The workload defines the consumers of the resources. In the context of this report a workload is composed of jobs, defined as a collection of computational tasks. Thus, a job j has n_j tasks $T^j_1, \ldots, T^j_{n_j}$.
- **Resources**: Resources, required to execute the workload, consist of a set of *distributed* nodes or computers, with one or more processing cores, *connected* by a, typically high-speed, network. These resources may be organized in computing clusters in a local environment or in widely distributed and scalable data centers [9]. Resources are assumed to be able to execute any type of computational task and consist of whole computing units, with main memory, storage devices and access to networks. We assume that nodes can only communicate by exchanging messages through the network.
- **Requirements**: The scheduling requirements determine the scheduling goal and other requirements that must be met by the solution. Typically, the scheduling goal is to optimize one or a combination of performance metrics related to the scheduling decisions. Another important scheduling requirement is the scheduling level. It determines the granularity or the level of detail considered when making a scheduling decision. We consider two levels of scheduling decisions: *job* and *task*[3].

In practice, the scheduling activity involves dynamic components. Both workload and resources may vary with time. In order to model these dynamic aspects, we consider $\mathcal{T} \subseteq \mathbb{R}^+$ to denote the set of time instants of interest, which may be discrete or continuous. At any time $t \in \mathcal{T}$ the workload is composed by a set $\mathcal{W}_t$ of jobs. At any time $t \in \mathcal{T}$ the resources are composed by a set $\mathcal{R}_t$ of interconnected computing nodes. The set $\mathcal{R}_t$ may change with time because resources may fail, may become unavailable for maintenance, or new resources may become available. Nevertheless, there are static properties of the workload and/or resources that do not change over time.

Let $\mathcal{W}$ and $\mathcal{R}$ represent the static aspects of the workload and resources respectively; they are properly discussed in the next section. Let $\mathcal{Q}$ be the set of scheduling requirements that must be satisfied, including the scheduling goal pursued. We define a scheduling problem $\mathcal{SP}$ as a tuple $(\mathcal{W}, \mathcal{R}, \mathcal{Q})$. A scheduling problem may be stated as the following optimization problem.

$$\begin{aligned} &\text{Optimize} && \mathcal{Q}.\text{goal} \\ &\text{Subject to} && \\ & && \text{Static features of } \mathcal{W} \\ & && \text{Static features of } \mathcal{R} \\ & && \text{Requirements in } \mathcal{Q} \end{aligned}$$

A scheduling solution $\mathcal{SS}$ must always be associated with a given scheduling problem. There may be more

[2] bit.ly/dssarchive

[3] Each task consists of one or more (lightweight) processes that must be scheduled at the computing node assigned to run the task. This constitutes a third level of scheduling, i.e., process level. Process level scheduling is related to the processes and threads execution at the operating system level and is outside the scope of this report.

than one solution to the same scheduling problem. These solutions may be compared in terms of their performance in solving the problem.

3 Taxonomy of scheduling in distributed systems

The taxonomy proposed here is organized into two parts, one that characterizes a scheduling problem and another that characterizes a scheduling solution.

The part of the taxonomy related to the problem consists of 17 static features that fall into three groups: workload ($\mathcal{W}$), resources ($\mathcal{R}$) and scheduling requirements ($\mathcal{Q}$). We summarize the possible values of each static feature in Figure 1.

3.1 Workload description

Seven features characterize the static aspect of the workload $\mathcal{W}$.

1. *$\mathcal{W}1$ - Job source*. Defines if jobs come from *multiple users* or a *single user* and if the workload consists of *multiple-jobs* or a *single-job*. Reasonable combinations are: *single user/single-job*, *single-user/multi-job* and *multi-user/multi-job*. When the workload comes from many users, scheduling is often performed from the provider standpoint.

2. *$\mathcal{W}2$ - Job structure*. Defines the allowed number of tasks per job and the dependency relations and communication needs among the tasks. First, this feature defines if jobs are multi-task or single-task. For multi-task jobs, one has to determine the homogeneity of the tasks. Tasks are homogeneous when they require similar resource demands and are heterogeneous otherwise. The many tasks of a job may have precedence constraints and communication needs to be satisfied, in which case they are *dependent*. Dependency between tasks often brings to the scheduling problem the challenge of data locality, since data transfers may come at a cost. When there are neither precedence relations among the tasks nor communication needs, tasks are *independent*. Based on this discussion, the job structure may be: *single-task*, *independent homogeneous multi-task*, *independent heterogeneous multi-task*, *dependent homogeneous multi-task* or *dependent heterogeneous multi-task*. The trivial case of a workload that is both single-job and single-task is not interesting and is not considered in this report.

3. *$\mathcal{W}3$ - Job flexibility*. *Rigid* jobs require a fixed quantity of resources and cannot execute on fewer or more resources. This quantity is defined by the user at job submission time. There are, however, other classes of jobs: moldable, malleable and evolving jobs [10]. When a *moldable* job starts, some entity, possibly, a scheduler, decides on the quantity of resources to provide the job at submission time. This quantity cannot be reconfigured during the job execution. *Malleable* jobs are moldable jobs whose computing requirements can change during execution by the scheduler or other system entity. Finally, *evolving* jobs are similar to malleable jobs, but the user is the one who decides, on the fly, about the quantity of resources to assign to the job.

4. *$\mathcal{W}4$ - Arrival process*. Determines the set of jobs considered by the scheduler when making scheduling decisions. In an *open* workload model, jobs come to the system at any time and leave the system after being executed, i.e., the number of jobs in the system is not constant. In a *closed* workload, the number of jobs to be scheduled is fixed.

5. *$\mathcal{W}5$ - Workload composition*. This feature is determined by the *programming model*, which drives the kinds of relationships that must hold between the tasks of a job. Some examples include embarrassingly parallel jobs, in which all tasks are independent from one another, and MapReduce jobs, in which all map tasks must finish before the reduce tasks start execution. A workload may be formed by jobs that follow the *same* programming model or may be *heterogeneous*. A workload that consists of jobs of the same programming model may be classified as:

   - *Same model/homogeneous*, when jobs are similar in terms of structure, number of tasks and in terms of demands required;
   - *Same model/same structure*, when jobs are similar in terms of structure, number of tasks but differ in terms of demands required;
   - *Same model/diverse*, when jobs use the same programming model but are different in terms of structure, number of tasks, and demands required. For instance, a workload that consists of many different MapReduce jobs is considered diverse because the number of maps and reduces differ among jobs as well as the demands of each task.

   Dependence relations and communication patterns do not exist if jobs are single-task. As a consequence, when the workload consists of multiple jobs of a single-task, the workload composition must be *same model/homogeneous* or *same model/same structure*.

6. *$\mathcal{W}6$ - Quality of service*. Jobs that form the workload may be associated to service level agreements (SLAs). When SLAs are violated penalties may be imposed. These jobs are *SLOs aware*, since they require service level objectives (SLOs) to be met. Jobs

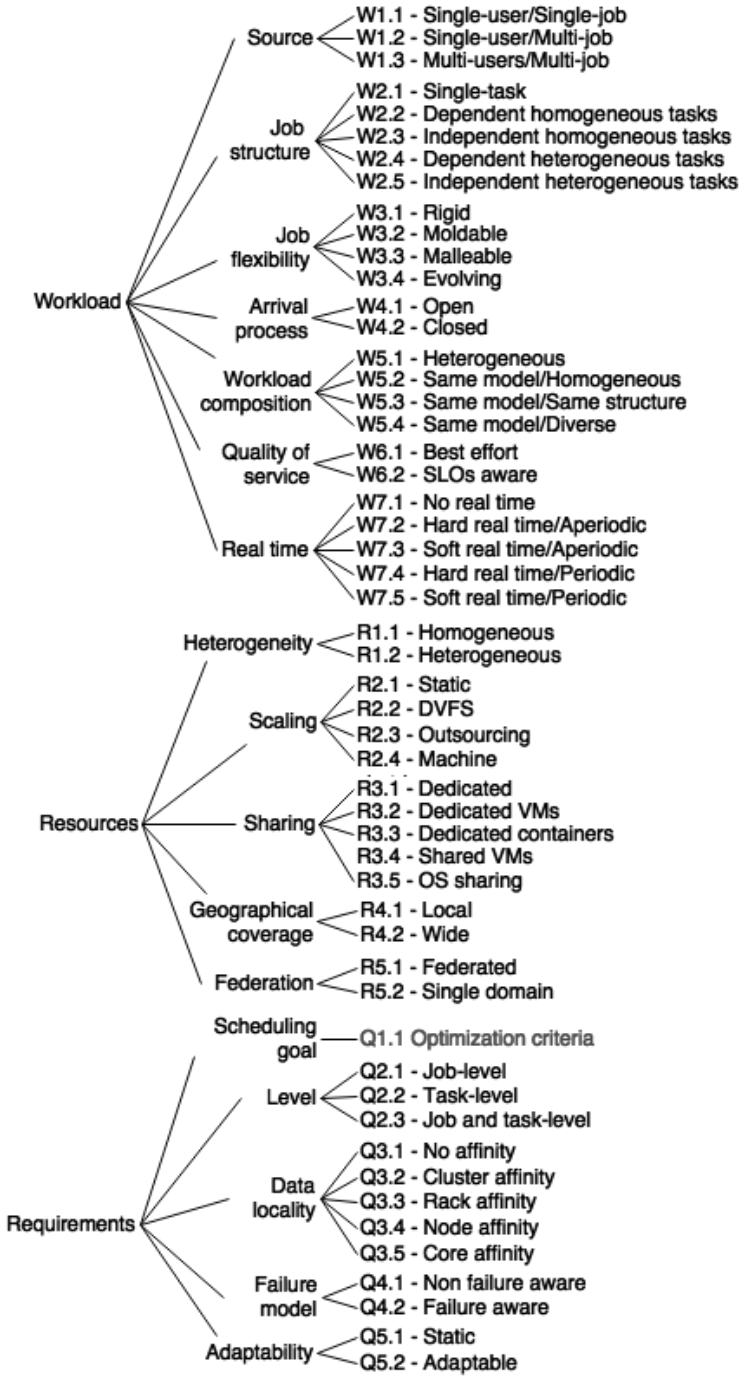

Figure 1: Summary of static features related to a scheduling problem.

that are not associated to SLAs are considered *best effort* jobs.

7. *$\mathcal{W}7$ - Real time*. The workload may consist of real time jobs or non real time jobs. For the former case, we distinguish between real time jobs with *hard deadlines* and *soft deadlines*. We also consider whether tasks are periodic or non-periodic. When the workload is *hard* or *soft real time*, it is necessarily *SLOs aware*.

3.2 Resource description

We identified five static features that characterize distributed resources.

1. *$\mathcal{R}1$ - Resource heterogeneity*. Individual nodes that form the cluster may be *heterogeneous* or *homogeneous*. Homogeneous clusters consist of similar nodes in terms of processing power, storage, and networking capabilities. Heterogeneous clusters are made up of nodes with different computing powers, in terms of processing, storage, or communication speeds.

2. *$\mathcal{R}2$ - Resource scaling*. The scheduler can see the resources it can use as a fixed or dynamic infrastructure in terms of processing capacity. Some infrastructures allow rapid capacity changes in response to variations in the workload. The total capacity of a *fixed-capacity* resource platform does not vary in the short term. On the other hand, some distributed systems allow *dynamic scaling*. Three common situations lead to dynamically scalable infrastructures:

    - *Shutdown resources*: some nodes are turned off in order to save energy. This temporarily reduces the online capacity of the infrastructure. But the total capacity is rapidly restored by turning on the machines;
    - *Outsourcing*: it is possible to rapidly acquire resources from other resource providers, such as infrastructure as a service (IaaS) providers or grid peers;
    - *DVFS*: Dynamic Voltage and Frequency Scaling has been recently explored by schedulers that target energy savings. Modern processors and disks may work in different voltage or power modes with a corresponding performance trade-off [11]. When the resource nodes respond to dynamic voltage changes, they are energy conserving and such responses change the overall capacity of the infrastructure.

    Dynamic resource scaling allows the scheduler to consider a wide spectrum of operating points that imply in different energy consumption and different costs for running the infrastructure.

3. *$\mathcal{R}3$ - Sharing*. Sharing determines how a job's performance may be influenced by other jobs. When a job runs in a *fully dedicated* node, the job is isolated and cannot influence the performance of other jobs. Another possibility is to have each job running on a dedicated virtual machine (VM) on top of a physical node. Virtualization technology enables resource sharing among different user processes inside the same physical node with minimum interference [12]. In the case of *dedicated VMs*, physical resources may still be shared among different jobs running on other VMs on the same physical resource. It is also possible to have many jobs sharing the same virtual machine. In this case, resources are *shared VMs*. Another possibility is to use *dedicated containers* that isolate processes from one another, such as Linux control groups [13]. The fourth possibility is to have resources shared without virtualization or other technique, which we call *OS sharing*. In this case, the tasks of different jobs run on the same physical node coordinated by the operating system. When the workload is *single-job*, there is only one job running and, consequently, there is no resource sharing among jobs. Thus, in this case, resources are necessarily *dedicated*;

4. *$\mathcal{R}4$ - Geographical coverage*. A cluster may be classified into *local* or *wide* depending on the geographically coverage of its nodes. The topology and bandwidth connecting them is an important attribute for wide-area clusters.

5. *$\mathcal{R}5$ - Federation*. A federation is the union of several smaller parts that perform a common action as in grid computing [14, 15, 16]. *Federated* resources are shared among different administrative domains in a coordinated fashion. Different domains may not only lead to different resources configurations, but also impose different usage rules. Federation brings new challenges to scheduling in terms of security, geographical issues, and the opportunistic usage of idle resources. When all the nodes are under the control of the same entity, they are *non-federated*. A federated cloud is an association of cloud providers for the deployment and management of multiple cloud computing services to match business needs.

3.3 Scheduling requirements

As discussed above, the solution of a scheduling problem requires that a set of requirements $\mathcal{Q}$, listed below, be satisfied.

1. *$\mathcal{Q}1$ - Scheduling goal*. The goal of a scheduling solution is to maximize or minimize the value of a metric of interest. Typical examples include minimize makespan, maximize resource utilization, maximize throughput, meet deadlines, and minimize energy

consumption. Sometimes there may be a need to optimize several, and often times conflicting, metrics. For example, users may want to minimize individual job completion times while providers may want to maximize resource utilization. Trade-offs such as this are often dealt by means of utility functions [17, 18], which allow for multi-criteria optimization.

2. *Q2 - Scheduling level*. Determines whether jobs and/or tasks are being scheduled:
   - *Job-level only* scheduling: decides which job will run next. Jobs not selected to run may be placed in a waiting queue or may be rejected to be resubmitted.
   - *Task-level only* scheduling: decides which task of a given job will run and in which node;
   - *Job and task level* scheduling: the solution considers both job and task level scheduling decisions.

   Figure 2 illustrates the operation of the two scheduling levels. First, submitted jobs are selected by an admission controller, which is a job-level scheduler that uses a scheduling policy $\mathcal{P}_J$. This scheduler determines the jobs that will gain access to the resources next. The admitted jobs become the active jobs or the active workload. The task-level scheduler sees only the active workload and assigns tasks to resources according to a task-level policy $\mathcal{P}_T$. Both job and task-level policies may be informed by the state of resources in order to make better decisions. These levels are not mutually exclusive and may be used together.

3. *Q3 - Data locality*. Some scheduling policies consider the effect of data locality, which can be classified as: *cluster affinity*, *rack affinity*, *node affinity* and *core affinity*. With *no affinity*, schedules are made independently for each task of a job, regardless of data transfer requirements. Cluster affinity is important for clusters that cover a wide geographical area to ensure that a task is executed by a node within a subset of nodes that has most of the data and binaries required to run the task (e.g., [19, 20]). With rack affinity, the scheduler tries to schedule tasks at processors in the rack where the input data is stored (e.g., [21, 22]). Node affinity means that tasks should run in any of the processor cores of a given node (e.g., [23, 24]). Finally, with core affinity tasks must be scheduled at a specific processor core to benefit from cache state.

4. *Q4 - Failure model*. A scheduling solution may be *failure-aware* or *non failure-aware*. In the first case, the scheduling solution considers that a cluster's nominal capacity may change over time due to node failures; thus, task reassignments may be required.

5. *Q5 - Adaptability*. A solution to a scheduling problem, given by the policies $(\mathcal{P}_J, \mathcal{P}_T)$, is *adaptable* if at least one of these policies is allowed to change in response to variations in the workload or resources.

3.4 Similarity between two scheduling problems

A well defined taxonomy for scheduling problems allows us to systematically compute the similarity, and consequently the dissimilarity, between two problems. The similarity can be determined based on the values of each static feature of two given problems and is defined in the interval $[0, 1]$. Values near 1 indicate that the two problems are very similar and values near 0 mean that the two problems are very different. The similarity $S_{1,2}$ between two problems $\mathcal{SP}_1$ and $\mathcal{SP}_2$ is defined through the following simple matching coefficient:

$$\begin{aligned} S_{1,2} &= \frac{\text{\# matching features}}{\text{\# features}} \\ &= \frac{1}{|\mathcal{W}| + |\mathcal{R}| + |\mathcal{Q}|} \times \\ &\Big(\sum_{f \in \mathcal{W}} sm_{1,2}(f) + \sum_{f \in \mathcal{R}} sm_{1,2}(f) + \sum_{f \in \mathcal{Q}} sm_{1,2}(f) \Big) \end{aligned}$$

where $sm_{1,2}(f)$ is one if scheduling problems $\mathcal{SP}_1$ and $\mathcal{SP}_2$ share the same feature f and zero otherwise. The sum of the cardinalities of $\mathcal{W}$, $\mathcal{R}$ and $\mathcal{Q}$ indicates the total number of features that describe a problem. The dissimilarity between the two problems is given by $1 - S_{1,2}$.

3.5 Features of scheduling solutions

According to [2], the scheduling solution $\mathcal{SS}$ is used to efficiently and effectively manage the access to and use of the set of resource by its various consumers. We extend their ideas to define a taxonomy for scheduling solutions in distributed systems. A scheduling solution $\mathcal{SS}$ may be characterized according to:

1. *S1 - Optimality*. A scheduling solution may be *optimal* or *sub-optimal*. Optimal solutions are achieved through *mathematical programming* [22, 25]. Sub-optimal solutions may apply different techniques including *mathematical programming* (e.g., [26, 27]), *multi-agent systems* (e.g., [28]), *economic-based* [29, 30] and *combinatorial search* (e.g., [21, 23, 31]).

2. *S2 - Operation*. A scheduler may be *online* or *offline*. An offline scheduler makes scheduling decisions based on a complete knowledge of the entire workload and infrastructure. On the other hand, an online scheduler does not know the future and scheduling decisions are made in response to events

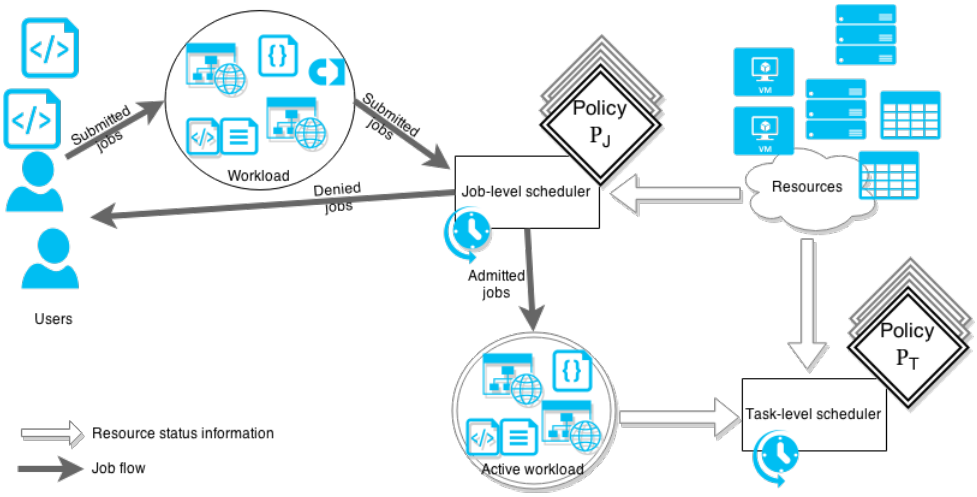

Figure 2: Components of scheduling process.

(e.g., job arrivals and departures, task completions, and node failures).

3. *S3 - Topology distribution*. This feature characterizes the location of the components that participate in the scheduling decision making process. At a high level, these components may be physically *distributed* or *non-distributed*. Distributed components may be:

   (a) *Centralized* or *decentralized*. The scheduling solution is *decentralized* when more than one component of the scheduler has the authority to make scheduling decisions (e.g., [20, 32]). Alternatively, when the authority to make scheduling decisions is centralized in one component, the scheduler is said to be *centralized* (e.g., [21, 23, 33]).

   (b) Regarding tasks assignments, physically distributed schedulers may apply the *pull-based* model or the *push-based* model [34]. In the pull-based model, resources pull tasks from a task repository associated with the global scheduler. In the push-based model, the global scheduler pushes tasks towards the workers, depending on the workers' preferences and capacities.

   There are, thus, five possibilities concerning topology distribution: *non distributed*, *distributed centralized pull-based*, *distributed centralized push-based*, *distributed decentralized pull-based* and *distributed decentralized push-based*.

4. *S4 - Flexibility*. A scheduler may be *rigid* or *flexible*. Flexible schedulers accept schedule changes on the fly and may be characterized by the following non exclusive behaviors:

   (a) Migratory: This property defines if the scheduler is *migration aware* or *non migration-aware*. Migration-aware schedulers may migrate a running task from one processor to other processor. Non migration-aware schedulers do not migrate running tasks;

   (b) Preemptive: Tasks may be temporarily interrupted and later resumed by *preemptive* schedulers. *Non-preemptive* schedulers do not interrupt tasks already running.

   A rigid scheduler does not admit changes on the schedule once tasks start to run. Thus, running tasks may not migrate or be preempted. Therefore, regarding flexibility, a scheduler may be *rigid*, *flexible migratory-aware non-preemptive*, *flexible non migratory-aware preemptive*, *flexible migratory-aware preemptive* or *flexible non migratory-aware non-preemptive*. The latter case represents flexible solutions that use other techniques different from migration and preemption. It is reasonable to establish that *offline* schedulers are necessarily *rigid*.

We summarize the static features of a scheduling solution $\mathcal{SS}$ in Figure 3.

4 Review Protocol

The following research questions served as the basis for our analysis of parallel job scheduling in distributed systems.

1. *RQ1*: How to define a scheduling problem in the context of parallel job scheduling in distributed platforms?

2. *RQ2*: How to classify scheduling solutions in the context of parallel job scheduling in distributed platforms?

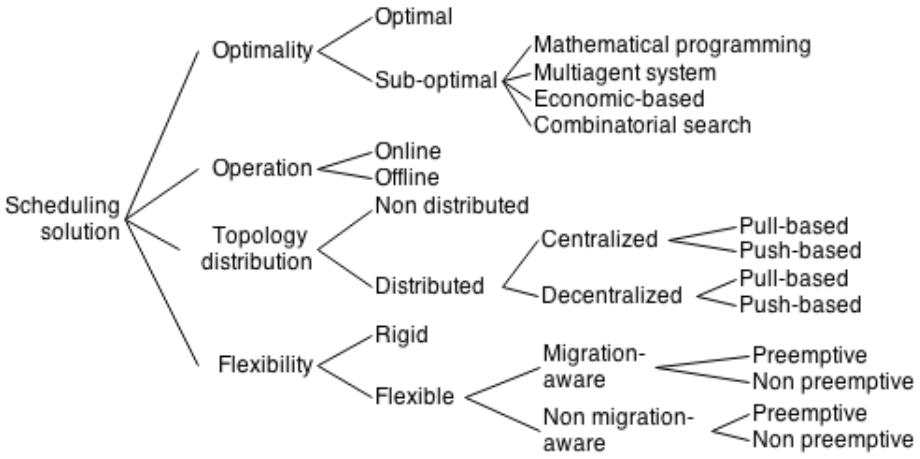

Figure 3: Summary of static features related to a scheduling solution.

3. *RQ3*: What are the most popular scheduling problems being investigated in the past ten years in the context of parallel job scheduling in distributed platforms?

4. *RQ4*: What are the most popular classes of scheduling solutions proposed in the past ten years in the context of parallel job scheduling in distributed platforms?

The first two questions are answered by the taxonomy proposed above. Questions RQ3 and RQ4 are answered in Section 6. To answer these questions, we collected papers about job scheduling in distributed systems. We read these papers and classified their scheduling problems and solutions using the taxonomy proposed here.

The methodology we used to collect the papers followed the general guidelines for performing systematic literature reviews in software engineering proposed by the Software Engineering Group at University of Keele [35]. The process includes the identification of the research questions (presented above) and a review protocol to be followed (see details on the protocol in Appendix A).

The following sources were considered: ACM Digital Library, IEEExplorer, ScienceDirect and SpringerLink. We searched for papers whose titles contain terms that satisfied the following criteria:

(scheduling $\vee$ schedule $\vee$ allocation $\vee$ scheduler) *and* (cloud $\vee$ grid $\vee$ cluster $\vee$ "data center") *and* (job $\vee$ application $\vee$ DAG $\vee$ MapReduce $\vee$ hadoop $\vee$ task $\vee$ workflow)

We also collected a second set of papers using the following search strategy: highly cited papers (i.e., more than 8 citations per year since publication), obtained from Google scholar [4], not found during the previous search, and that satisfy the following relaxed search criteria: (scheduling $\vee$ schedule $\vee$ allocation $\vee$ scheduler) *and* (cloud $\vee$ grid $\vee$ cluster $\vee$ "data center" $\vee$ job $\vee$ application $\vee$ DAG $\vee$ MapReduce $\vee$ hadoop $\vee$ task $\vee$ workflow). We call these papers the set of *GS papers*.

Papers to be collected must satisfy the above criteria and must have been published from January 1^{st}, 2005 to May, 1^{st}, 2015.

We manually collected 1,050 papers, of which 52 are GS papers as defined above. The data collection occurred in two moments: April, 28^{th} to May, 1^{st} and June, 15^{th}, 2015 [5]. The following information was gathered from each paper: year of publication, title, authors, source of publication, number of citations. Information regarding number of citations was collected manually from Google Scholar.

The average number of citations per year of each paper (CPY_{paper}) is computed according to the following formula. The numerator, NC_{paper}, is the total number of citations received by the paper and the denominator, $Y_{NC} - Y_{paper}$ is an indication of the number of years since publication estimated as the difference between the year the number of citations was collected (i.e., 2015) and the year the paper was published.

$$CPY_{paper} = \frac{NC_{paper}}{Y_{NC} - Y_{paper}} \quad (1)$$

We sort all the papers in descending order of number of citations per year and read the most cited papers in order to classify the scheduling problems and solutions according to our taxonomy until 100 problems are classified.

We recognize that the number of citations of a paper is not constant throughout the years but the same metric was applied to all papers. We treated all papers the same with respect to Equation 1 regardless of the month

[4] https://scholar.google.com

[5] At first, we have not considered the search term "workflow", but after reading some papers we realized that it would be very important to include this word. Papers collected in June, 15^{th} include this term.

of their publication during the year even though we recognize that this may cause some slight variability.

Another aspect worth considering in our survey protocol is our choice of keywords. This choice restricts us to papers that explicitly match the search criteria in their the titles. Of course, it is possible that some very relevant papers were left out from this process. One such example is the paper that presents the LATE scheduler for MapReduce applications [36]. Although our search criteria are very strict and may miss a few papers, it guarantees that the papers found are actually very relevant to the topic. This is especially important for the automated analysis we conducted and report in Section 5.

During the classification process of 100 problems we analyzed the 123 most cited papers and 25% (i.e., 31 papers) of them were considered out of scope. Papers considered out of scope were related to VM placement or scheduling of parallel jobs in one multi-processor computer; or they are reduced versions of other papers that present the same problems/solutions. From these 31 papers that are out of scope, 18 papers were GS papers. The error rate of the non GS papers is thus 16%. Since the great majority of the 1,050 papers satisfy all the search criteria, we can consider that this is approximately the error rate of the set of all papers analyzed.

5 General statistics on Scheduling Papers

This section presents general statistics related to the set of 1,050 papers. Most of these papers are related to grid scheduling, followed by papers related to cloud and clusters as can be seen in the second column of Table 1. The category indicated by "Unknown" in the table (and all the figures that follow) represents GS papers whose titles do not explicitly indicate the resource category considered.

Table 1: Frequencies of resource categories

Resource categories	All papers	Popular papers	Top-7
Cloud	27.0%	30.5%	3
Cluster	17.1%	17.6%	2
Data center	1.6%	4.6%	1
Grid	50.9%	28.2%	0
Unknown	3.4%	19.1%	1

Figure 4 presents the resource categories of the papers through the years. As the figure shows, the number of papers published per year increases over time. In fact, a regression analysis indicates a linear increase with an adjusted R^2 equal to 0.83. We predict with 95% of confidence that the number of papers published with titles that fit our search terms in 2015 is between 130 and 166. The increase in the number of papers reinforces the

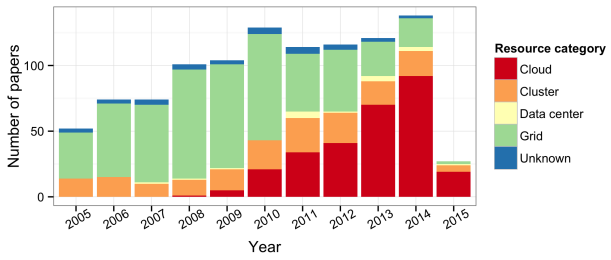

Figure 4: Number of papers published through the years and categories of resources.

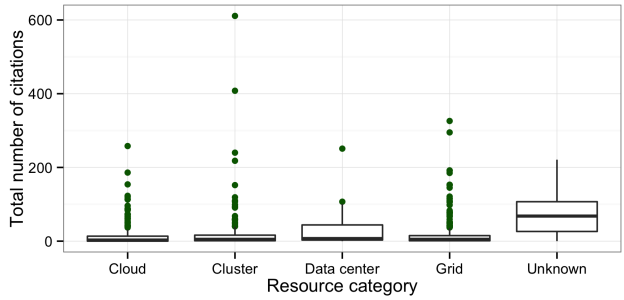

Figure 5: Box-plots of citations grouped by resource category.

importance of a well defined taxonomy to allow for new research in scheduling to be built on top of the prior art.

An analysis of the figure and underlying data also shows that the number of grid-related papers peaked in 2008 and started to decrease linearly with an adjusted R^2 of 0.86 (the 95% CI of the slope is $[-19.0, -6.5]$). The number of grid papers in 2015 is comparatively very low. The figure clearly demonstrates the emergence of the cloud computing paradigm starting in 2009. Since then, the number of papers related to clouds has increased linearly with an adjusted R^2 of 0.96 (the 95% CI of the slope is $[11.2, 19.0]$). The rate of increase of cloud papers is greater than the overall increase rate of papers of all resource categories. In fact, since 2013, the number of papers related to scheduling in cloud environments exceeds the number of papers related to scheduling in all other distributed environments. We predict with 95% confidence that the number of papers published in 2015 related to scheduling in cloud computing will be between 80 and 115.

We now analyze the popularity of job scheduling papers in distributed systems. The results are shown in Fig. 5, which presents box-plots of the number of citations grouped by resources categories.

As expected, the largest average citation number is for the "Unknown" category, which refer to GS papers. The two most cited papers are both related to clusters, they are the Hadoop Fair Scheduler [21] and Quincy [22]. It is clear from the box-plots that, on average, the number

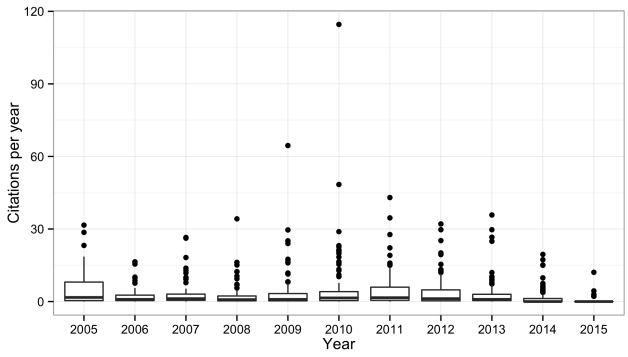

Figure 6: Box-plots of citations per year grouped by year.

of citations each paper receives is very low. On average, papers are not cited more than 5 times (median) in total. 21.5% of the papers have no citations at all. 9.15% of the papers are cited once and 9.05% are cited twice. The proportions of papers cited three, four and five times are 4.5%, 4.6% and 3.8%, respectively. Less than half of the papers (47.4%) are cited more than five times and most of the papers (52.6%) received at most five citations.

The number of papers that are rarely cited is increasing over the years. Again, a regression analysis indicates a linear increase with an adjusted R^2 of 0.8. The slope of this linear function is 9.7, on average greater than the slope of the linear function that considers all the papers (8.3). However, the 95% confidence intervals of the slopes overlap[6] and it is not possible to state with 95% confidence that one exceeds the other. However, there is a clear trend that more and more papers will be published and, among them, even more papers will be barely cited.

Once more, this situation reinforces the need for a taxonomy so that researchers can easily identify, among the huge number of scheduling papers, those that are relevant to their specific research. Addressing this problem will improve the overall quality of the research in the area and will increase the number of citations to previously published papers.

Figure 6 depicts the popularity of papers according to the number of citations computed by Equation (1).

On average, the rate of citations per year does not exceed one (median), which means that half the papers are cited at most once a year. More specifically, 21.5% of the papers were never cited, 29.0% of them are cited at most once per year, 15.2% of the papers were cited more than once and at most twice per year, 8.5% of the papers were cited more than twice and at most three times per year, and 3.9% of the papers were cited more than three times and at most four times per year. The remaining 21.9% of the papers were cited more than four times per year.

[6]These intervals are $[6.1, 13.3]$ for the less popular papers and $[5.4, 11.2]$ for all papers.

We performed a Kruskal Wallis Test, followed by pairwise Wilcoxon Rank Tests[7] to determine if the number of citations per year changes according to the year the paper was published. We excluded papers from 2015 since we do not have a full year worth of papers. Our analysis showed that the average number of citations per year is statistically the same for all papers published from 2005 to 2013. Additionally, we can say with 95% of confidence that the papers published in 2014 have a lower number of citations per year than those published previously. We did a similar analysis excluding the most popular papers and we obtained the same results.

Figures 5 and 6 show that some papers are outliers being far more cited than the average. More precisely, we considered a popular paper as an outlier whose number of citations per year is greater than the third quartile plus $1.5 \times IQR$ where IQR is the interquartile range. That implies that 131 papers are outliers (i.e., 12.5% of all the papers). Together, these well-cited 131 papers are responsible for 64.0% of all citations.

Figure 7 presents the number of popular papers published throughout the years as well as the categories of resources considered by these papers. It is not visually clear anymore the decreasing number of grid papers and the increasing number of cloud computing papers. The proportion of papers for each resource category considered by the most popular papers is presented in the third column of Table 1. Even among the most cited papers, there are outliers. The average number of citations per year for the top-7 cited papers exceeds 33.5. The fourth column of Table 1 shows a breakdown of these papers by resource category.

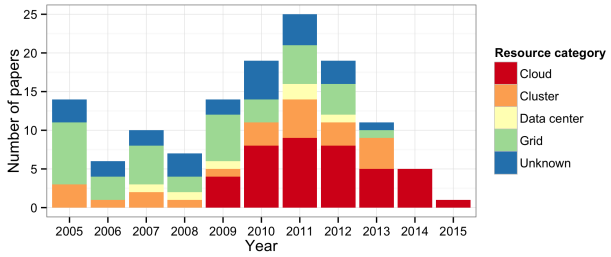

Figure 7: Number of popular papers published through the years and categories of resources considered.

6 The top-100 scheduling problems and solutions

This section presents features of 100 scheduling problems and solutions extracted from 92 papers that, among the papers we collected, are the most cited per year

[7]We used these non-parametric versions of ANOVA and t-test because the data are not normally distributed.

over the last ten years. These problems are listed in Appendix B. The taxonomy presented previously is used to characterize the scheduling problems and solutions. Table 2 presents the distribution of problems per resource category. Together, computational grids and cloud computing, both outsourced infrastructures, surpass cluster environments in terms of number of problems studied considering each category.

Table 2: Resource categories considered by the top-100 problems.

Resource category	Number	Frequency
Cluster	39	38.6%
Grid	33	32.7%
Cloud	29	28.7%

After classifying 100 scheduling problems we realized that the period we considered is fortunately very rich in four aspects because it includes:

1. key years regarding research related to scheduling on computational grids, including the golden years and the decreasing popularity of this resource category;
2. the year in which MapReduce programming was introduced and the subsequent surge in MapReduce scheduling research (13% of the problems are related to MapReduce jobs);
3. the advent of cloud computing, which unveiled new scheduling problems on virtualized distributed infrastructures; and
4. the era of huge data centers and the increasing concern to reduce their energy costs (this was addressed by 16% of the problems).

Figure 8 shows the distribution of scheduling problems related to each resource category from 2005 to 2015. The most productive period in terms of research impact was from 2009 to 2012, which contribute with 60.4% of the published problems classified. In terms of source, 35% of the problems were published by IEEE and 32% by Elsevier. From these publishers, the journals that contributed the most papers were Elsevier's Future Generation Computer Systems Journal with 15% and IEEE Transactions on Parallel and Distributed Systems with 5%.

We analyze now the most frequent values for each feature. In general, most of the problems (71%) consider workloads composed of multiple rigid jobs from multiple users. The job arrival process may be closed or open; there is no typical value. Workflow jobs (dependent heterogeneous multi-task) are the most typical types of jobs appearing in 51% of the problems. The second most typical kind of job is single-task (20%), followed by bags of

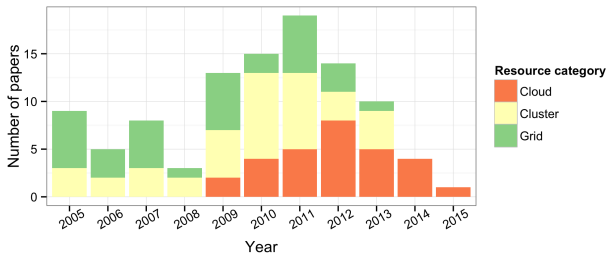

Figure 8: Distribution of top100 scheduling problems through the years and categories of resources considered.

tasks jobs (independent heterogeneous multi-task) (14%). Regarding the composition of the workload, three values are more common, but no typical value covers the majority of the problems: heterogeneous jobs (38%), same model/diverse (36%) and same model/same structure (25%). A vast majority (82%) of the problems do not require service level guarantees and 95% of the problems have a non real time workload.

Heterogeneous resources are considered by 76% of the problems. Indeed, heterogeneity is usually considered in both resources and workloads, which makes scheduling problems more challenging. Regarding resource scaling, outsourcing appears in 52% of the problems (those related to grids and cloud) and 35% of the problems consider a static set of resources. The resources are typically dedicated to a given job (34%) or VMs are dedicated to run the jobs (49%). There is no consensus about the geographical coverage of the resources; they are local in 55% of the problems. Non-federated environments are more common and appear in 65% of the problems. Data locality is not a requirement in 79% of the problems and 76% of them are non failure aware. Regarding the scheduling goal, makespan is still the most important scheduling criterion: 60% of the problems minimize makespan and possibly other metrics of interest. Finally, almost all problems do not require adaptable scheduling solutions but do require job and task level scheduling.

Regarding the solutions, there is no much variation among the groups. Only four solutions are optimum [22, 25, 37, 38]. Most of the solutions (85%) use a combinatorial search method to find sub-optimal schedules. A wide variety of optimization techniques and different new heuristics are applied in general. Biologically inspired methods are very often used, such as particle swarm [39, 40], ant colony optimization [41, 42] as well as improvements on traditional algorithms such as List based scheduling [43, 44] or min-min/min-max [41, 33, 19]. Genetic algorithms are also common [45, 46, 47]. Some other sub-optimal solutions are found by means of mathematical programming [30, 20, 48] or multi-agent systems [49, 40]. Finally others use both mathematical programming and combinatorial search solutions [50, 26, 27]. A very large number (79%) of the solu-

tions are online and, surprisingly, 90% of them are rigid. Indeed, offline solutions are always rigid, but online solutions have the opportunity to correct wrong scheduling decisions on the fly, which it is still being done with parsimony by 10% of the solutions. Thirty five percent of the solutions are not distributed, and half of the solutions are distributed centralized push-based; 15% of the solutions are decentralized and are more concentrated on grid-related papers.

In order to study these one hundred problems, we applied a hierarchical clustering mechanism to extract the features of typical groups of problems. Eight groups were identified, the results and analysis of each group are presented in Subsection 6.1. In Subsection 6.2 we present a validation study that illustrates the confidence we have in the classification presented here and points to some features that are often not mentioned by the authors of the classified papers.

6.1 Groups of problems (and their solutions)

We carried out an agglomerative hierarchical clustering analysis to identify meaningful subgroups of scheduling problems studied in the past decade. These groups are formed in a hierarchical fashion, based on the similarities of each of the features[8] from the 100 problems classified using our taxonomy. We used the distance metric presented in subsection 3.4. It is up to the analyst to decide the number of groups to identify. We conducted an analysis to determine a good balance between cluster separation (based on inter-cluster distance) and cluster definition (based on intra-cluster distance). We concluded that eight groups provided the best clustering in terms of group uniformity. Table 3 presents quantitative information about each group as well as the labels chosen to identify the groups, based on their most popular features.

Figure 9 illustrates the characteristics of each group. The graph uses the numbers in Figure 1 to identify the values of each category. For instance, value 1 for *Source* indicates "Single-user/single-job" and value 2 for *Heterogeneity* indicates "heterogeneous" resources. The darkest lines and points indicate where most of the values of the groups are concentrated. Similarly, the more transparent the points and lines, the less frequent that value appears in the group.

A great majority of scheduling problems consider rigid jobs, scheduling is done at both job and task level and the solution is not adaptable.

[8]We excluded the scheduling goal from the features because it does not have previously determined values. However, we could find some patterns regarding scheduling goals inside the groups.

6.1.1 Group 1: Failure and data locality aware scheduling of workflow jobs in clusters

The first group consists of eleven scheduling problems. Table 4 shows detailed information on the features of these problems. Some representative problems of this group are the Hadoop Fair Scheduler [21], Quincy [22] and Flex [51]. The problems in this group consider workloads with multiple jobs that can arrive at any time from many users. The jobs are multi-task and the tasks are dependent and usually heterogeneous. Typically, the workload is composed of diverse rigid jobs of the same model. Indeed, 55% of the problems in this group consider MapReduce jobs. Finally, the workload does not require service level guarantees and is not real time. Typically, these problems consider resources that cover a local geographic area, are non federated, and are static. These are typical features of clusters, and, as a matter of fact, most of the problems (82%) consider a cluster as the resource category. Resources are often heterogeneous. Variations regarding resource scaling come from the GreenHadoop [23], which shuts down machines to save energy, and Purlieus [52], which considers an outsourced cloud environment. Resources are typically shared at the operating system level. A common feature to almost all problems is that they require some kind of affinity. Only three problems do not require the exploration of data locality [53, 54, 55]. Finally, all the problems require failure-aware solutions.

The solutions proposed for these problems are online schedulers with the following properties: (i) they are sub-optimal and apply a combinatorial search solution; (ii) they follow a distributed centralized push-based architecture; and (iii) they are rigid (no migration nor preemption).

6.1.2 Group 2: Scheduling of single-task jobs in heterogeneous dedicated clusters

The second group consists of seven scheduling problems (see Table 5 for their features). Some representative problems of this group are SCINT and FCFS-Backfill-XInt [56][9], as well as GAS [57]. The problems in this group usually consider that the workload has multiple rigid jobs that can arrive at any time from many users. All the problems consider single-task jobs. Typically, the workload consists of jobs of the same model that can be homogeneous or follow the same structure. Additionally, the workload does not require service level guarantees and is not real time. All the problems in this group consider heterogeneous resources that are non-federated and cover a local area; typical characteristics of clusters. There is no typical value related to scaling, but 68% of them apply DVFS or shut down machines to save energy. In terms of sharing, resources are usually dedicated to run a given job. None of the problems require the ex-

[9]Both problems are presented in the same paper.

Table 3: Quantitative information about the groups identified.

Grp.	Size	Freq.	Group names
1	11	10.9%	Failure and data locality aware scheduling of workflow jobs in clusters
2	7	6.9%	Scheduling of single-task jobs in heterogeneous dedicated clusters
3	10	9.9%	Scheduling of workflow jobs in homogeneous shared clusters
4	16	15.8%	Scheduling of workflow jobs in dedicated clusters
5	6	5.9%	QoS-driven scheduling of hard real time jobs in homogeneous clusters
6	25	24.8%	Scheduling of single-task and bag of tasks jobs in grids
7	13	12.9%	Scheduling of heterogeneous multi-task jobs in federated resources
8	13	12.9%	Scheduling of single-user workflow jobs in cloud environments

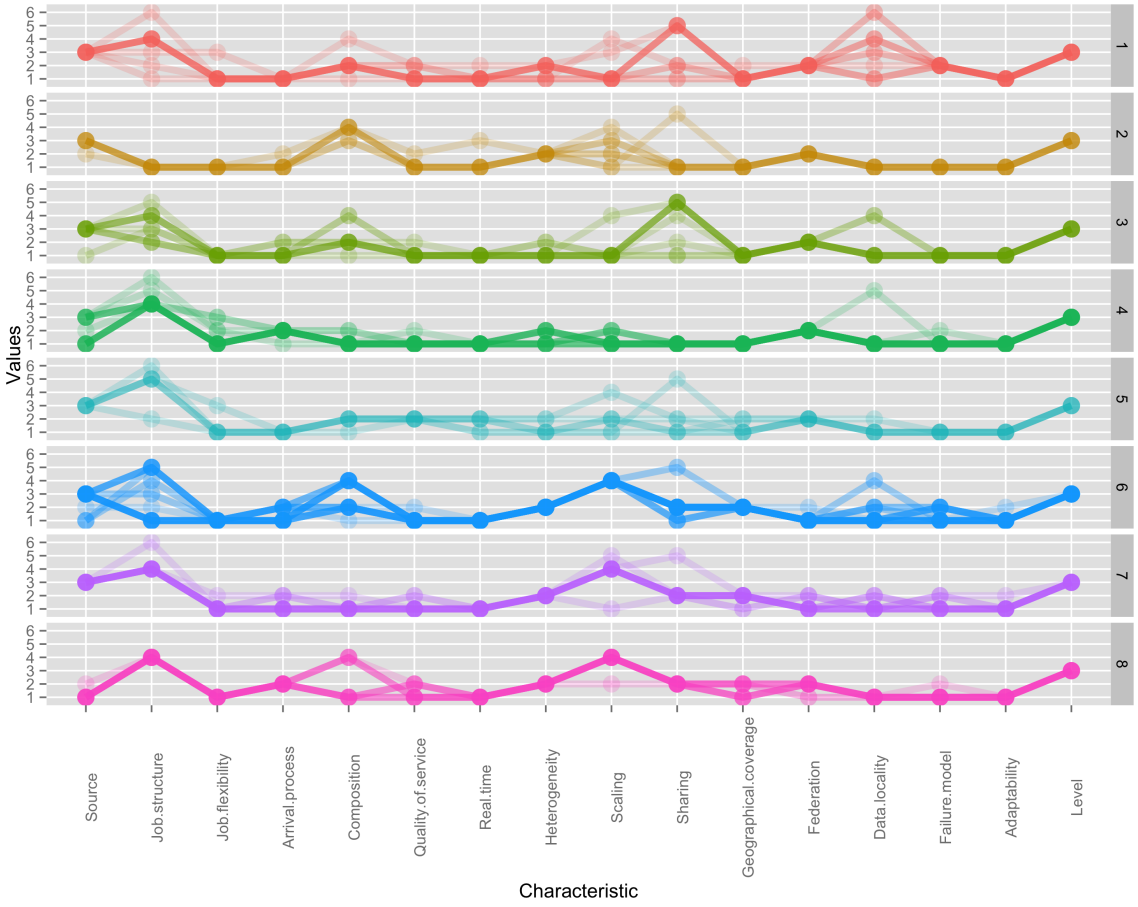

Figure 9: The eight groups of problems.

Table 4: Features of group 1 - Failure and data locality aware scheduling of workflow jobs in clusters.

Feature	**Value**	**Number**	**Frequency**
Source	*Multi-user/Multi-job*	11	100%
Job structure	(In)Dependent heterogeneous multi-task	1	9%
Job structure	*Dependent heterogeneous multi-task*	7	64%
Job structure	Dependent homogeneous multi-task	1	9%
Job structure	Independent heterogeneous multi-task	1	9%
Job structure	Single-task	1	9%
Job flexibility	*Rigid*	10	91%
Job flexibility	Malleable	1	9%
Arrival process	*Open*	11	100%
Composition	Heterogeneous	1	9%
Composition	*Same model/diverse*	9	82%
Composition	Same model/same structure	1	9%
Quality of service	*Best effort*	8	73%
Quality of service	SLOs aware	3	27%
Real time	*No real time*	10	91%
Real time	Hard real time	1	9%
Heterogeneity	*Heterogeneous*	7	64%
Heterogeneity	Homogeneous	4	34%
Scaling	Outsource	1	9%
Scaling	Shutdown machines	1	9%
Scaling	*Static*	9	82%
Sharing	Dedicated VMs	3	27%
Sharing	Dedicated	2	18%
Sharing	*OS sharing*	6	55%
Geographical coverage	*Local*	10	91%
Geographical coverage	Wide	1	9%
Federation	*Non federated*	11	100%
Data locality	No affinity	3	27.3%
Data locality	Cluster affinity	1	9%
Data locality	Node/rack affinity	2	18.2%
Data locality	Rack affinity	2	18.2%
Data locality	Node affinity	3	27.3%
Failure model	*Failure aware*	11	100%
Adaptability	*Fixed*	11	100%
Level	*Job and task level*	11	100%

ploration of data locality nor failure awareness. Energy savings seems to be important to this class of problems, appearing as part of the scheduling goal in 57% of the problems.

The solutions proposed for these problems are all sub-optimal and use combinatorial search solutions. These solutions typically have the following properties: (i) they are online schedulers; (ii) they are non distributed; and (iii) they generate rigid schedules.

6.1.3 Group 3: Scheduling of workflow jobs in homogeneous shared clusters

The third group consists of ten scheduling problems (see details on features in Table 6). Some representative problems of this group are ARIA [58], BFS [29], Pred [59] and Min/Max [33]. Most of the problems consider that the workload has multiple jobs that can arrive at any time from many users. The jobs typically consist of many tasks with dependency relations. Typically, the workload is composed of diverse rigid jobs of the same model. Indeed, 70% of the problems in this group explicitly identify jobs as MapReduce jobs or workflow jobs in the title of the papers. The workloads usually do not require service level guarantees and are not real time. Apart from [33] and [60], all other problems consider homogeneous resources. All these problems consider non-federated resources covering a local geographic area. In terms of scaling, resources are mainly static. Again, these are common features of clusters, and, as a matter of fact, most of the problems (60%) identify a cluster as the resource category (the other 40% are related to cloud environments). Resources are typically shared at the operating system level or they are shared through virtualization technology. Only two problems explore data locality, both at the node level [61, 62]. Finally, the problems in this group do not require a failure aware scheduling solution.

The solutions proposed for these problems are online schedulers that typically have the following properties: (i) they are sub-optimal and use a combinatorial search solution; (ii) they follow a distributed centralized push-based architecture; and (iii) they are rigid (no migration nor preemption).

6.1.4 Group 4: Scheduling of workflow jobs in dedicated clusters

The fourth group consists of sixteen scheduling problems (see Table 7 for their features). Some representative problems of this group are the Probabilistic Backfilling [31], XInt [26], CPGA/TDGA [45], LDCP [63] and PALS/PATC [43]. There is no typical value for the workload source, which may come from multiple users that submit multiple jobs, or may be a single job from a single user. These problems consider a fixed set of jobs to schedule, which characterizes the arrival process as closed. The jobs are usually dependent heterogeneous multi-task, typically modeled by DAGs, which leads to heterogeneous workloads. In terms of flexibility, most of the problems consider rigid jobs, although malleable [64] [10] and moldable [31, 65] jobs are also seen. The workloads of these problems do not require service level guarantees and are not real time. There is no typical value regarding heterogeneity, almost half of the problems consider heterogeneous resources and a little over half consider homogeneous resources. All the problems in this group consider resources that are dedicated to run a job, cover a local geographic area and are non federated. Most of the problems consider a static resource infrastructure, but some problems consider a dynamic infrastructure in which resources have DVFS [44, 43, 66]. 81.3% of the problems consider a cluster as the resource category. Makespan is an important metric to optimize in most of the problems in this group (69%). Finally, almost all problems in this group are non failure aware and do not explore data locality.

The solutions proposed for these problems are non distributed, offline, rigid and sub-optimal, applying a combinatorial search solution.

6.1.5 Group 5: QoS-driven scheduling of hard real time jobs in homogeneous clusters

The fifth group consists of six scheduling problems (see feature details in Table 8). Some representative problems of this group are TTS [27], GMCE/GMP [67] and BIP [25]. These problems consider that the workload has multiple jobs that can arrive at any time from many users. The jobs are typically multi-task, and in most of them, tasks are heterogeneous and independent. Typically, the workload is composed of diverse rigid jobs of the same model. The workload to be scheduled requires a certain level of service, since it is typically real time, except from the two problems presented in [68]. Although the workloads of these problems are not explicitly real time, they require some level of quality of service and consider meeting job deadlines as the scheduling goal. Typically, these problems consider homogeneous resources that cover a local geographic area and are non federated. Regarding resource sharing, half of the problems consider dedicated physical resources, and two problems consider dedicated VMs. Most of the problems pursue the minimization of some sort of cost, which may be related to energy costs or total execution cost. The only problem that does not explicitly tries to minimize some kind of cost is TTS [27], whose goal is to maximize throughput and meet deadlines. There is no typical value regarding scaling, but half the problems consider dynamic scaling with DVFS. Most of the problems do not require the exploration of data locality and do not require failure awareness. Finally, it is interesting to note that although

[10] Two problems are presented in [64]

Table 5: Features of group 2 - Scheduling of single-task jobs in heterogeneous dedicated clusters.

Feature	**Value**	**Number**	**Frequency**
Source	*Multi-user/Multi-job*	6	86%
Source	Single-user/Multi-job	1	14%
Job structure	*Single-task*	7	100%
Job flexibility	*Rigid*	7	100%
Arrival process	Closed	1	14%
Arrival process	*Open*	6	86%
Composition	Same model/homogeneous	2	29%
Composition	*Same model/same structure*	5	71%
Quality of service	*Best effort*	6	86%
Quality of service	SLOs aware	1	14%
Real time	*No real time*	6	86%
Real time	Soft real time	1	14%
Heterogeneity	*Heterogeneous*	7	100%
Scaling	Outsource	1	14.2%
Scaling	Dynamic/DVFS	2	28.6%
Scaling	Shutdown machines	2	28.6%
Scaling	Static	2	28.6%
Sharing	OS sharing	1	14%
Sharing	*Dedicated*	6	86%
Geographical coverage	*Local*	7	100%
Federation	*Non federated*	7	100%
Data locality	*No affinity*	7	100%
Failure model	*Non failure aware*	7	100%
Adaptability	*Fixed*	7	100%
Level	*Job and task level*	7	100%

Table 6: Features of group 3 - Scheduling of workflow jobs in homogeneous shared clusters.

Feature	**Value**	**Number**	**Frequency**
Source	*Multi-user/Multi-job*	9	90%
Source	Single-user/Single-job	1	10%
Job structure	Dependent heterogeneous multi-task	4	40%
Job structure	Dependent homogeneous multi-task	3	30%
Job structure	Independent homogeneous multi-task	2	20%
Job structure	Independent heterogeneous multi-task	1	10%
Job flexibility	*Rigid*	10	100%
Arrival process	Closed	2	20%
Arrival process	*Open*	8	80%
Composition	Heterogeneous	1	10%
Composition	*Same model/diverse*	7	70%
Composition	Same model/same structure	2	20%
Quality of service	*Best effort*	9	90%
Quality of service	SLOs aware	1	10%
Real time	*No real time*	10	100%
Heterogeneity	Heterogeneous	2	20%
Heterogeneity	*Homogeneous*	8	80%
Scaling	Outsource	1	10%
Scaling	*Static*	9	90%
Sharing	Shared VMs	1	10%
Sharing	Dedicated VMs	1	10%
Sharing	Dedicated	2	20%
Sharing	*OS sharing*	6	60%
Geographical coverage	Local	10	100%
Federation	*Non federated*	10	100%
Data locality	*No affinity*	8	80%
Data locality	Node affinity	2	20%
Failure model	*Non failure aware*	10	100%
Adaptability	*Fixed*	10	100%
Level	*Job and task level*	10	100%

Table 7: Features of group 4 - Scheduling of workflow jobs in dedicated clusters.

Feature	**Value**	**Number**	**Frequency**
Source	Multi-user/Multi-job	7	44%
Source	Single-user/Multi-job	1	6%
Source	Single-user/Single-job	8	50%
Job structure	(In)dependent heterogeneous multi-task	1	6%
Job structure	*Dependent heterogeneous multi-task*	14	88%
Job structure	Independent heterogeneous multi-task	1	6%
Job flexibility	*Rigid*	12	75%
Job flexibility	Malleable	2	12.5%
Job flexibility	Moldable	2	12.5%
Arrival process	*Closed*	15	94%
Arrival process	Open	1	6%
Composition	*Heterogeneous*	14	87.5%
Composition	Same model/diverse	2	12.5%
Quality of service	*Best effort*	15	94%
Quality of service	SLOs aware	1	6%
Real time	*No real time*	16	100%
Heterogeneity	Heterogeneous	7	44%
Heterogeneity	*Homogeneous*	9	56%
Scaling	Dynamic/DVFS	4	25%
Scaling	*Static*	12	75%
Sharing	*Dedicated*	16	100%
Geographical coverage	*Local*	16	100%
Federation	*Non federated*	16	100%
Data locality	*No affinity*	15	94%
Data locality	Core affinity	1	6%
Failure model	Failure aware	1	6%
Failure model	*Non failure aware*	15	94%
Adaptability	*Fixed*	16	100%
Level	*Job and task level*	16	100%

all the 100 problems require job and task level scheduling, the problems in this group present more sophisticated job level scheduling, which perform admission control actions.

The solutions proposed for these problems are online schedulers that typically have the following properties: (i) they are sub-optimal and use a combinatorial search solution; (ii) they follow a distributed centralized push-based architecture; and (iii) they generate rigid schedules.

6.1.6 Group 6: Scheduling of single-task and bag of tasks jobs in grids

The sixth group is the largest with twenty five scheduling problems (see Table 9 for details on features). Some representative problems in this group are min/max/suff(CTT) [19], FDPSO [39], BACO [41], ACO [69] and StorageAffinity [70]. Most of the problems consider that the workload has multiple jobs from many users. But some problems consider the scheduling of a single-job from a single-user (e.g., min/max/suff(CTT) [19] and Swarm-opt [50]). 80% of the jobs consist of a single task or many independent heterogeneous tasks, or bag of tasks applications. Typically, the workload consists of an open arrival process of jobs of the same model, which may be diverse or of the same structure. Finally, the workload does not require service level guarantees and is not real time. All the problems in this group consider that resources are heterogeneous, outsourced and cover a wide area. Except from one problem, all other problems consider federated resources. As a matter of fact, these are the main characteristics of computational grids [11] [16]. This group contains one of the two problems among the 100 problems that requires adaptability [73]. Indeed, heterogeneous federated and widely distributed resources make up a very dynamic environment that can benefit from adaptation. Resource sharing occurs mainly through virtualization in the form of dedicated VMs that run the jobs. Some problems require the exploration of data locality at the cluster level, but most of them do not have data locality requirements. Most of the problems in this group do not require failure awareness. Finally, makespan is the metric or one of the metrics to be optimized in 68% of the problems in this group.

The solutions proposed for these problems are online schedulers and typically have the following properties: (i) they are sub-optimal and apply a combinatorial search solution or multi-agent system; (ii) they follow a distributed centralized push-based architecture; and (iii) they are rigid (no migration nor preemption).

[11] Some problems in this group define the infrastructure as a kind of federated cloud [50, 71, 72], which, in our view, is a resource category very similar to computational grids.

6.1.7 Group 7: Scheduling of heterogeneous multi-task jobs in federated resources

The seventh group consists of thirteen scheduling problems (see Table 10 for their features). Some representative problems in this group are the DMDP [30], DCLS/AMMS(EL) [20], SAT/PSOE [37], and MOTS [32]. The problems in this group usually consider that the workload has multiple rigid jobs that can arrive at any time from many users. All the problems consider workloads composed by many dependent heterogeneous tasks, often modeled as DAGs, which leads to heterogeneous workload compositions. Additionally, the workloads do not require service level guarantees and are not real time. All the problems in this group consider that resources are heterogeneous and most of them consider outsourced, federated, and widely distributed resources, which are the typical characteristics of grids and federated clouds. Resources are typically shared as dedicated VMs. Most problems do not require the exploration of data locality and are not failure-aware. Minimization of the makespan is part of the scheduling criteria in 85% of the problems. One problem in this group requires adaptability in order to deal with federated and heterogeneous resources [42].

The solutions proposed for these problems are often online schedulers with the following properties: (i) they are sub-optimal and use a combinatorial search solution; (ii) they follow a distributed centralized or decentralized push-based architecture; and (iii) they generate rigid schedules, although preemption is applied in 31% of the solutions.

6.1.8 Group 8: Scheduling of single-user workflow jobs in cloud environments

The eighth group consists of thirteen scheduling problems (Table 11 shows their features). The investigation of these problems is mainly concentrated in the period that ranges from 2011 to 2014. Cloud computing platforms appeared in other groups, but this is the group in which cloud infrastructures are more prevalent (seen in 69%). It is worth mentioning that in some papers an environment very similar to a cloud environment is referred to as utility grid (e.g. [74, 75, 40]). Some representative problems of this group are VOO [76], HybridGA [77], HCOC [78] and PSO [79]. These problems consider that the workload is fixed as single rigid jobs that come from a single user. The job is always a workflow, which is a dependent heterogeneous multi-task job. The workload is not real time and typically does not require service level guarantees. The problems in this group consider a heterogeneous environment in which physical outsourced resources are shared as dedicated VMs. These features are as expected for a group that represents scheduling in cloud environments because the scheduler is usually able to acquire different flavors of virtual machines from the cloud providers. Two distinct approaches are seen

Table 8: Features of group 5 - QoS-driven scheduling of hard real time jobs in homogeneous clusters.

Feature	**Value**	**Number**	**Frequency**
Source	*Multi-user/Multi-job*	6	100%
Job structure	(In)Dependent heterogeneous multi-task	1	17%
Job structure	Dependent homogeneous multi-task	1	17%
Job structure	*Independent heterogeneous multi-task*	4	67%
Job flexibility	*Rigid*	5	83%
Job flexibility	Malleable	1	17%
Arrival process	*Open*	6	100%
Composition	Heterogeneous	1	17%
Composition	*Same model/diverse*	5	83%
Quality of service	SLOs aware	6	100%
Real time	*Hard real time*	4	67%
Real time	No real time	2	33%
Heterogeneity	Heterogeneous	2	33%
Heterogeneity	*Homogeneous*	4	67%
Scaling	Outsource	1	17%
Scaling	Dynamic/DVFS	3	50%
Scaling	Static	2	33%
Sharing	Dedicated VMs	2	33%
Sharing	Dedicated	3	50%
Sharing	OS sharing	1	17%
Geographical coverage	*Local*	4	67%
Geographical coverage	Wide	2	33%
Federation	*Non federated*	6	100%
Data locality	*No affinity*	5	83%
Data locality	Cluster affinity	1	17%
Failure model	*Non failure aware*	6	100%
Adaptability	*Fixed*	6	100%
Level	*Job and task level*	6	100%

Table 9: Features of group 6 - Scheduling of single-task and bag of tasks jobs in grids.

Feature	**Value**	**Number**	**Frequency**
Source	*Multi-user/Multi-job*	20	80%
Source	Single-user/Single-job	4	16%
Source	Single-user/Multi-job	1	4%
Job structure	Dependent heterogeneous multi-task	2	8%
Job structure	Independent heterogeneous multi-task	8	32%
Job structure	Dependent homogeneous multi-task	1	4%
Job structure	Independent homogeneous multi-task	2	8%
Job structure	Single-task	12	48%
Job flexibility	*Rigid*	25	100%
Arrival process	Closed	10	40%
Arrival process	*Open*	15	60%
Composition	Heterogeneous	1	4%
Composition	Same model/diverse	12	48%
Composition	Same model/same structure	12	48%
Quality of service	*Best effort*	24	96%
Quality of service	SLOs aware	1	4%
Real time	*No real time*	25	100%
Heterogeneity	*Heterogeneous*	25	100%
Scaling	*Outsource*	25	100%
Sharing	*Dedicated* VMs	18	72%
Sharing	Dedicated	5	20%
Sharing	OS sharing	2	8%
Geographical coverage	*Wide*	25	100%
Federation	Non federated	1	4%
Federation	*Federated*	24	96%
Data locality	*No affinity*	18	72%
Data locality	Cluster affinity	5	20%
Data locality	Node affinity	2	8%
Failure model	Failure aware	9	36%
Failure model	*Non failure aware*	16	64%
Adaptability	*Fixed*	24	96%
Adaptability	Adaptable	1	4%
Level	*Job and task level*	25	100%

Table 10: Features of group 7 - Scheduling of heterogeneous multi-task jobs in federated resources.

Feature	**Value**	**Number**	**Frequency**
Source	*Multi-user/Multi-job*	13	100%
Job structure	(In)Dependent heterogeneous multi-task	1	8%
Job structure	*Dependent heterogeneous multi-task*	12	92%
Job flexibility	*Rigid*	12	92%
Job flexibility	Moldable	1	7%
Arrival process	*Open*	10	77%
Arrival process	Closed	3	23%
Composition	*Heterogeneous*	12	92%
Composition	Same model/diverse	1	8%
Quality of service	*Best effort*	11	85%
Quality of service	SLOs aware	2	15%
Real time	No real time	13	100%
Heterogeneity	Heterogeneous	13	100%
Scaling	*Outsource*	11	84.6%
Scaling	Outsource and shutdown machines	1	7.7%
Scaling	Static	1	7.7%
Sharing	*Dedicated VMs*	12	92%
Sharing	OS sharing	1	8%
Geographical coverage	Local	2	15%
Geographical coverage	*Wide*	11	85%
Federation	*Federated*	10	77%
Federation	Non federated	3	23%
Data locality	*No affinity*	10	77%
Data locality	Cluster affinity	3	23%
Failure model	Failure aware	3	23%
Failure model	*Non failure aware*	10	77%
Adaptability	*Fixed*	12	92%
Adaptability	Adaptable	1	8%
Level	*Job and task level*	13	100%

regarding geographical coverage of the resources. In one approach, there are different cloud providers widely spread and the scheduler may acquire resources from all of them [76, 75, 74]. In the other approach, there is a single cloud provider from which resources are acquired [80, 81]. Regardless of the approach, the resources are typically non-federated because acquired resources can be used as needed. Almost all the problems pursue the minimization of cost. Besides, they are all concerned with decreasing the makespan or with meeting jobs deadlines. None of the problems require the exploration of data locality and most of them do not pursue failure awareness.

The solutions proposed for these problems are all sub-optimal and use combinatorial search solutions that typically generate rigid schedules. These solutions are mainly online (61%) and they may be non distributed or may follow a distributed centralized push-based architecture (46%).

6.2 Classification confidence

The process of classifying one hundred problems was not always easy. We did it based on the papers that describe the problems and the solutions. Unfortunately, some features are not explicitly mentioned. In these cases, the confidence level of the inferred value may not be high. Sometimes, even though the feature is not mentioned, it is possible to infer its value with high confidence by reading the paper, especially the evaluation results and the scenarios tested. However, in other occasions, their values are inferred with medium confidence. We identify these situations in order to know the features that are neglected the most by the authors when describing their problems and solutions. Figure 10 presents the confidence achieved when classifying each feature of the scheduling problems and solutions.

Sharing is the characteristic less mentioned in the papers. This is an important feature to be considered when describing a scheduling problem because it directly affects the solution. A solution for resources that are shared among many jobs at the operating system level may possibly be different from the solution when resources are dedicated to run tasks of the same job without sharing.

Surprisingly, the second characteristic that was sometimes a factor of low confidence is the source of the workloads. Some papers do not mention users, or customers, or clients and many do not clearly define the context in which the scheduler's solution will be used and the jobs are submitted. The other feature that is hardly mentioned is the arrival process of the jobs. Although we could infer their values, the feature is usually neglected. Finally, the topology distribution of the solutions is not mentioned in many papers, in which cases we usually considered the solution as not distributed.

We hope that from now on all the papers that propose solutions to scheduling problems in distributed systems use the taxonomy we propose. By so doing, the process of classification is facilitated and consequently it will easier for researchers to find similar problems and solutions to the ones they are investigating.

7 Related work

Parallel job scheduling in distributed systems has been a fertile research ground for some decades. Thus, the number of papers with solutions and surveys related to the area is enormous. We consider here two classes of related work: seminal work that inspired the taxonomy proposed in this report and surveys that aim at exploring the state-of-the-art on scheduling in distributed systems. We position our taxonomy proposal in relation to other related taxonomies and we emphasize the aspects that make our survey unique in relation to other related surveys.

7.1 Inspirational taxonomies

Our taxonomy is logically organized into two parts: one describes scheduling problems and the other describes scheduling solutions. The scheduling problem taxonomy is mainly inspired on the scheduling problem theory presented in [3] and reviewed in some books [8, 7]. According to the theory, scheduling problems should be defined in terms of: (i) the processing environment α, (ii) the tasks β, and (iii) the optimality criterion γ. Our three groups, respectively, resources ($\mathcal{R}$), workload ($\mathcal{W}$) and requirements ($\mathcal{Q}$) come from that idea. Although the $\alpha|\beta|\gamma$ notation for deterministic scheduling problems has existed for decades, it is difficult and unnatural to apply it to scheduling problems in modern distributed systems because the notation misses important features to fully characterize scheduling problems in these systems. Workload structure, source, resource sharing, scaling, scheduling level and the adaptability requirement are a few examples of features that cannot be modeled by that notation.

The part of our taxonomy related to the description of scheduling solutions is mainly based on the taxonomy proposed by Casavant and Kuhl [2]. They defined a taxonomy to characterize scheduling solutions in general-purpose distributed computing systems. The main reason we think a revision of that taxonomy is important is that current applications and the environments in which they run became significantly more complex over the last thirty years. We expanded the Casavant and Kuhl taxonomy as follows:

- We introduced the notion of pull and push-based assignments, which is indeed more important when the resources set is big, and scalability issues are involved;

Table 11: Features of group 8 - Scheduling of single-user workflow jobs in cloud environments.

Feature	Value	Number	Frequency
Source	Single-user/Multi-job	1	8%
Source	*Single-user/Single-job*	12	92%
Job structure	*Dependent heterogeneous multi-task*	13	100%
Job flexibility	*Rigid*	13	100%
Arrival process	*Closed*	13	100%
Composition	*Heterogeneous*	8	61.5%
Composition	Same model/same structure	5	38.5%
Quality of service	*Best effort*	8	61.5%
Quality of service	SLOs aware	5	38.5%
Real time	*No real time*	13	100%
Heterogeneity	*Heterogeneous*	13	100%
Scaling	Dynamic/DVFS	1	8%
Scaling	*Outsource*	12	92%
Sharing	*Dedicated VMs*	13	100%
Geographical coverage	Local	6	46%
Geographical coverage	Wide	5	54%
Federation	*Non federated*	11	85%
Federation	Federated	2	15%
Data locality	*No affinity*	13	100%
Failure model	Failure aware	1	8%
Failure model	*Non failure aware*	12	92%
Adaptability	*Fixed*	13	100%
Level	*Job and task level*	13	100%

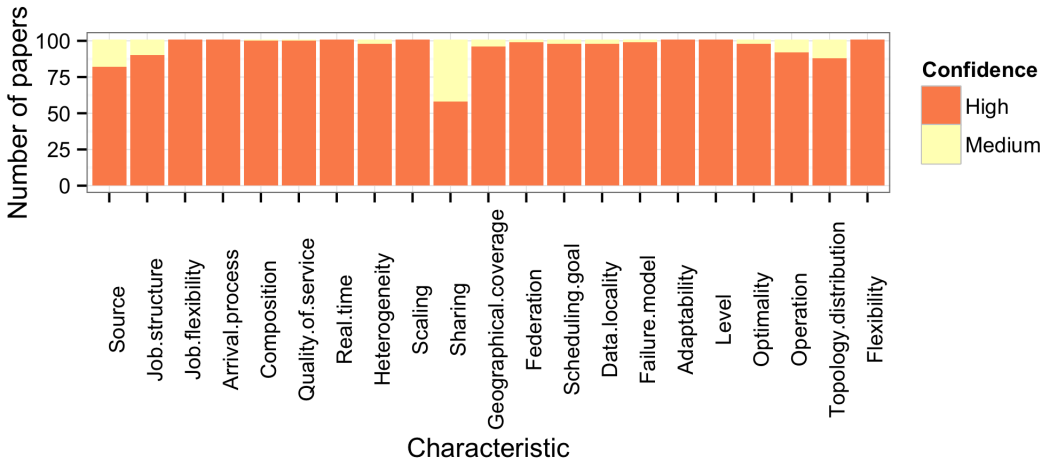

Figure 10: Classification confidence.

- Casavant and Kuhl considered more general techniques (e.g., (i) solution space enumeration and search, (ii) graph theoretic, (iii) mathematical programming, and (iv) queuing theoretic) for sub-optimal solutions. We replaced techniques (ii) and (iv) by multi-agent systems and economic based solutions. While queuing and graph theory are still useful, they are generally more appropriate to modeling the workload and resources, and the model may be used as part of a heuristic;
- We added one-time assignment and dynamic assignment as part of the taxonomy (see our flexibility feature). Casavant and Kuhl defined these as flat characteristics [12] not related to any specific branch of the taxonomy. We argue, however, that this is an important solution feature and we further subdivide it into different leaves to account for preemption, migration and other types of dynamic assignment;
- We also added adaptive and non-adaptive features (see our adaptability requirement). Casavant and Kuhl defined them as flat characteristics. We, however, consider this as a requirement of the scheduling problem, not a feature of the solution;
- For the sake of simplicity, we do not distinguish between sub-optimal and approximate solutions;

7.2 Related taxonomies

Many different yet sometimes overlapping taxonomies were proposed in the last ten years, each targeting a specific category of distributed platforms. There are taxonomies for scheduling in data grids [82, 83], computational grids [4, 84], cloud computing [5, 85] and utility-driven clusters [86]. All these taxonomies share the same problem: they are not general neither complete enough to characterize scheduling problems and solutions in distributed systems. To the best of our knowledge, a taxonomy that is broad and can be applied to any scheduling problem in distributed systems does not exist. Besides, they were proposed with the aim of surveying the state-of-the-art, and, unfortunately, they have not been used by researchers to describe their scheduling problems. At least, none of the papers classified used any kind of taxonomy to clearly define the scheduling problem being addressed.

Taxonomies for scheduling problems and solutions in computational grids were presented in [4]. They define five taxonomies: workflow model, scheduling criteria, scheduling process, resource model, and task model. These taxonomies do not explicitly separate problem and solution. It is important to separate problem from solution [8] because a taxonomy may be used to discover similar problems and to compare solutions to similar problems. It is worth mentioning that the taxonomies in [4] rely on the concept of a market of resources. Consequently, one of the optimization criteria is fixed: cost minimization. These five taxonomies cover a variety of dimensions, some of which are easy to generalize and are similar to some features in our taxonomies, such as criteria and workflow multiplicity, dynamism, resource diversity, migration, and workload component model. Even though many of the dimensions defined in [4] are relevant for describing scheduling problems and solutions for general distributed systems, they are not prepared to be used in the general context of any distributed platform.

Smanchat and Viriyapant [5] extended and complemented the grid taxonomies [4] for the cloud. The three taxonomies they proposed cover a variety of dimensions, all strongly related to the cloud environment. Two properties of their taxonomies are similar to scheduling criteria and task execution capacity we define in our taxonomy. Their taxonomies do not model scheduling requirements, except for scheduling goal. The scheduling solution is also not characterized by their taxonomies, except for the scheduling generation property, that defines how the scheduling decision is generated.

A taxonomy of market-based resource management systems for utility-driven cluster computing was proposed in [86] for scheduling solutions in which users assign utility to their jobs requests. The general goal is to maximize user's utility satisfaction. The authors organize the taxonomy into five sub-taxonomies. Among them, the resource model and the job model are quite related to our taxonomy of scheduling problems, but less detailed than ours. In spite of the similarities, the taxonomy proposed in [86] is not comprehensive enough to properly define scheduling problems and solutions.

Other taxonomies were presented but they do not cover the entire spectrum of scheduling problems. For example, [85] presented short taxonomies for inter-cloud and application brokering systems. These taxonomies characterize who owns the infrastructure, how the inter-cloud is constructed, who is responsible for application scheduling and the type of applications, the single similarity with our taxonomy. A taxonomy related to grids was presented in [84] for scheduling solutions only, classified as best-effort or QoS-constrained. Best effort in [84] refers to solutions that target makespan minimization without considering other goals or constraints while QoS constraint based scheduling aims at minimizing makespan under QoS constraints such as budgets or deadlines. We use similar terms, but with different meanings, to characterize the workload, not the solution. Moreover, the taxonomy in [84] characterizes solutions as heuristic and meta-heuristic based. Our taxonomy

[12] The other flat characteristics are not part of our taxonomy. They are: (i) load balancing, which we consider as a possible requirement of the scheduler; (ii) bidding, which is an economic-based technique to construct the schedule; and (iii) probabilistic, which is related to the quality of the solution and is already covered by the optimality feature.

covers an extended set of solution types with a higher level of specificity.

The authors in [87] defined a taxonomy of tools for dynamic task scheduling for high performance cluster computing. That taxonomy can assist tool designers and developers and is similar to the solution part of our taxonomy and to Casavant and Kuhl's taxonomy. The taxonomy in [87] encompasses four scheduling tool attributes, three of them similar to some of our features: (1) *target system*, which indicates resource heterogeneity and/or geographical coverage; (2) *control model*, which somehow resembles our distribution mode features; and (3) *scheduling strategy*, which is similar to policy operation. Despite some similarities, the taxonomy in [87] does not target the definition of scheduling problems or their solutions and is not readily usable for that.

Finally, a taxonomy for scheduling data-intensive applications in data grids is presented in [82, 83]. When data-intensive applications are being scheduled, it is reasonable to have a data management entity responsible for data transfers and data replication, which may be coupled or not to the scheduler. The requirements of the operation mode of this data management entity is not explicitly modeled by our taxonomy. A specific taxonomy must be used as a complement when necessary. However, our taxonomy indicates data-intensive operations by means of the *data locality* feature, which is important to characterize problems that require the exploration of data locality [21, 22, 23, 51].

7.3 Surveys on scheduling

This section discusses survey papers in the parallel job scheduling domain. However, none of them consider general distributed systems or classify the problems being addressed. Most of them address the solutions or consider a limited category of resources (e.g., grids or clouds).

In a status survey report on parallel job scheduling, the authors discuss the performance of the main scheduling approaches used in parallel supercomputers, clusters, and grids but not the definition of scheduling problems [88]. Research papers and commercial solutions are considered and described. No classification scheme is used neither a systematic review protocol.

Important concepts on grid computing scheduling problems and solutions using heuristic and metaheuristic approaches are described in [89]. They consider different grid models and for each model they identify the problem instance, which usually consists of workload and resources models. Different models to represent the workload and the resources are presented for each grid model. The goal of identifying typical problems and models is similar to our goal in this technical report, however they consider only grids and they do not define a clear taxonomy to be used. In that paper, the authors enumerate some differences between scheduling in general distributed systems and in grids. These differences are described below and are covered by our taxonomy.

1. Resources in a grid can join or leave the grid in an unpredictable way and there may be local policies that regulate resource usage. We use *federated* resources to model this feature. By defining that resources are federated, we indicate that resources are not under the control of the scheduler. Besides, we allow for shared resources to indicate that resources are not dedicated to run a single job;

2. High heterogeneity of jobs and resources. We can define resources as *heterogeneous*. Considering the workload, we have several features to identify workload heterogeneity, including *heterogeneous* jobs;

3. High heterogeneity of interconnection networks and the large scale of a grid. Resources that cover a *wide* geographical area are typically connected through the Internet using different interconnection networks.

4. The existence of local schedulers. We allow for two levels of scheduling: task and job level;

5. Security: we allow for resources to be shared through virtualization (*dedicated VMs*). It is reasonable to consider virtualization as a useful technique for coordinated and secure sharing of resources in grid computing. Security requirements may always be considered as a scheduling goal as in [42, 69, 53].

A survey on resource allocation in high performance distributed computing systems [90] considers generic distributed platforms, including cluster, grid, and cloud computing infrastructures. However, a formal, general taxonomy is not defined in [90]. Cluster, grid and cloud platforms are described in terms of some attributes, which somehow resemble some static features of our taxonomy and the taxonomies proposed in [86]. However, a different set of attributes is used to categorize each type of platform (cluster, grids, cloud). This is different from our work, in which the same taxonomy is used regardless of the resource platform. Moreover, their classification scheme conflates attributes related to scheduling solutions with those that characterize the resources and workload. Actually, there is no effort in [90] to characterize the specific scheduling problems being addressed and their focus is mostly on the solutions. Our work gives equal importance to problems and solutions.

Finally, other related surveys exist, each considering specific aspects of the scheduling solution. In [91], for instance, we find a literature review of scheduling solutions based on swarm intelligence and in [92] we find a survey on cloud workflow scheduling solutions.

8 Conclusions

This technical report proposes a taxonomy for parallel job scheduling in distributed systems. The taxonomy is organized into two parts: one that models the scheduling problem, considering the workload, the resources and the scheduling requirements; and other that models the scheduling solution. The taxonomy may be applied to any kind of distributed platform, including clusters, cloud computing infrastructures, and grids.

We then used a well-defined protocol to collect papers in which scheduling problems and solutions are described. 1050 papers were manually collected and an impact analysis was performed. We found that 12.5% of the papers are responsible for 64.0% of all citations and, unfortunately, almost one quarter of the papers are never cited. On average, papers are not cited more than 5 times during their whole life. These numbers confirm our hypothesis that research in this area rarely builds on prior art. Many features can be used to define a scheduling problem in this area. Changing the value of one feature may change the problem in a significant way. We classified 100 problems using our taxonomy and we found that most of the problems and their solutions are unique.

Following the impact analysis, we applied the taxonomy to classify one hundred problems and solutions. Based on this classification we carried out a hierarchical clustering analysis and found eight groups of problems that were investigated during the last ten years: (i) failure and data locality aware scheduling of workflow jobs in clusters, (ii) scheduling of single-task jobs in heterogeneous dedicated clusters, (iii) scheduling of workflow jobs in homogeneous shared clusters, (iv) scheduling of workflow jobs in dedicated clusters, (v) QoS-driven scheduling of hard real time jobs in homogeneous clusters, (vi) scheduling of single-task and bag of tasks jobs in grids, (vii) scheduling of heterogeneous multi-task jobs in federated resources, and (viii) scheduling of single-user workflow jobs in cloud environments.

The main challenge we dealt with during the design of the taxonomy was to identify features that were general enough to be included in the taxonomy and features that were too specific and should thus be excluded. As a result, we ended up with a general taxonomy; complementary taxonomies must be used if specific features need to be defined. Different resource models may lead to different complementary features. These complementary features may be borrowed by the specific taxonomies that already exist. As future work we will look into the organization of complementary taxonomies for very specific situations.

Drozdowski considers that devising a taxonomy for scheduling is a challenge and gives some recommendations [8], which are covered by our taxonomy. First, it pursues generality when describing the problem. We are quite confident about this aspect, especially after the classification of one hundred different problems. Second, a taxonomy must allow different scheduling levels, which we do model by the feature *level* in $\mathcal{Q}$. Third, a taxonomy must be specific when defining target applications. Indeed, we have a whole set of features ($\mathcal{W}$) to describe the workload to be scheduled. The taxonomy allows the classification of different job structures and workload compositions, giving the flexibility to model very different kinds of applications and workloads. Fourth, a taxonomy must be specific when describing the problem, taking into account the area of application of the scheduling algorithm and availability of input data. All the features that describe the resources and the workload characterize the area of application of the scheduling algorithm. Fifth, a taxonomy must consider communication aspects, which are implicit by the geographic coverage of the resources. Finally, the taxonomy must separate problem and solution, which was an important requirement we have pursued since the beginning of the taxonomy design.

Lastly, we published an online archive in which 100 problems and solutions are classified using the taxonomy. Our goal is to collaboratively maintain this archive, which allows researchers to submit a classification of a scheduling problem/solution. We hope that researchers will use the taxonomy and collaborate with the archive by submitting classifications of old and new scheduling problems. If authors use the taxonomy to classify their own work, the archive will be not only updated but also reliable. It is already possible to obtain a CSV file with all archived classifications and manipulate the data using statistical software. The archive also allows for the automatic selection of problems with certain feature values, which can be an important tool for those who want to know the state-of-the-art in a given scheduling niche.

Acknowledgements

Raquel Lopes would like to thank Mason's Department of Computer Science and its C4I Center for hosting her as a visiting scholar. Raquel Lopes visit is supported by CNPq/Brazil (grant 203418/2014-0). The authors would like to thank Marcus Carvalho for his help on the multi-variate graphics presented in this report.

References

[1] Dror G. Feitelson and Larry Rudolph. Towards convergence in job schedulers for parallel supercomputers. In *Proc. of the Wkshp. Job Scheduling Strategies for Parallel Processing*, IPPS '96, pages 1–26, London, UK, UK, 1996. Springer-Verlag.

[2] T. L. Casavant and J. G. Kuhl. A taxonomy of scheduling in general-purpose distributed com-

puting systems. *IEEE Trans. Softw. Eng.*, 14(2):141–154, February 1988.

[3] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G.Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In E.L. Johnson P.L. Hammer and B.H. Korte, editors, *Discrete Optimization II Proc. of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium co-sponsored by IBM Canada and SIAM Banff, Aha. and Vancouver*, volume 5 of *Annals of Discrete Mathematics*, pages 287 – 326. Elsevier, 1979.

[4] Marek Wieczorek, Andreas Hoheisel, and Radu Prodan. Towards a general model of the multi-criteria workflow scheduling on the grid. *Future Generation Computer Systems*, 25(3):237 – 256, 2009.

[5] Sucha Smanchat and Kanchana Viriyapant. Taxonomies of workflow scheduling problem and techniques in the cloud. *Future Generation Computer Systems*, (0):–, 2015.

[6] R. W. Hamming. One man's view of computer science. *J. ACM*, 16(1):3–12, January 1969.

[7] J. Błażewicz, K.H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz. *Handbook on Scheduling: From Theory to Applications*. Intl. Handbooks on Information Systems. Springer Berlin Heidelberg, 2007.

[8] Maciej Drozdowski. *Scheduling for Parallel Processing*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[9] Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, 2nd Edition*. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2013.

[10] Dror G. Feitelson and Larry Rudolph. Toward convergence in job schedulers for parallel supercomputers. In DrorG. Feitelson and Larry Rudolph, editors, *Job Scheduling Strategies for Parallel Processing*, volume 1162 of *Lecture Notes in Computer Science*, pages 1–26. Springer Berlin Heidelberg, 1996.

[11] Si-Yuan Jing, Shahzad Ali, Kun She, and Yi Zhong. State-of-the-art research study for green cloud computing. *The J. of Supercomputing*, 65(1):445–468, 2013.

[12] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. *SIGOPS Oper. Syst. Rev.*, 37(5):164–177, October 2003.

[13] Paul Menage, Paul Jackson, and Christoph Lameter. cgroups. `http://www.kernel.org/doc/Documentation/cgroups/cgroups.txt`, August 2012.

[14] Ian Foster, Carl Kesselman, and Steven Tuecke. The anatomy of the grid: Enabling scalable virtual organizations. *Intl. J. of High Performance Computing Applications*, 15(3):200–222, 2001.

[15] Walfredo Cirne, Francisco Brasileiro, Nazareno Andrade, LauroB. Costa, Alisson Andrade, Reynaldo Novaes, and Miranda Mowbray. Labs of the world, unite!!! *J. of Grid Computing*, 4(3):225–246, 2006.

[16] I. Foster, Yong Zhao, I. Raicu, and Shiyong Lu. Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Wkshp., 2008. GCE '08*, pages 1–10, Nov 2008.

[17] John Wilkes. *Utility Functions, Prices, and Negotiation*, pages 67–88. John Wiley & Sons, Inc., 2009.

[18] W. E. Walsh, G. Tesauro, J. O. Kephart, and R. Das. Utility functions in autonomic systems. In *Autonomic Computing, 2004. Proc. Intl. Conf. on*, pages 70–77, May 2004.

[19] Saurabh Kumar Garg, Rajkumar Buyya, and Howard Jay Siegel. Time and cost trade-off management for scheduling parallel applications on utility grids. *Future Generation Computer Systems*, 26(8):1344 – 1355, 2010.

[20] Jiayin Li, Meikang Qiu, Zhong Ming, Gang Quan, Xiao Qin, and Zonghua Gu. Online optimization for scheduling preemptable tasks on iaas cloud systems. *J. of Parallel and Distributed Computing*, 72(5):666 – 677, 2012.

[21] Matei Zaharia, Dhruba Borthakur, Joydeep Sen Sarma, Khaled Elmeleegy, Scott Shenker, and Ion Stoica. Delay scheduling: A simple technique for achieving locality and fairness in cluster scheduling. In *Proc. of the 5th European Conference on Computer Systems*, EuroSys '10, pages 265–278, New York, NY, USA, 2010. ACM.

[22] Michael Isard, Vijayan Prabhakaran, Jon Currey, Udi Wieder, Kunal Talwar, and Andrew Goldberg. Quincy: Fair scheduling for distributed computing clusters. In *Proc. of the ACM SIGOPS 22nd Symp. Operating Systems Principles*, SOSP '09, pages 261–276, New York, NY, USA, 2009. ACM.

[23] Íñigo Goiri, Kien Le, Thu D. Nguyen, Jordi Guitart, Jordi Torres, and Ricardo Bianchini. Greenhadoop: Leveraging green energy in data-processing frameworks. In *Proc. of the 7th ACM European Conference on Computer Systems*, EuroSys '12, pages 57–70, New York, NY, USA, 2012. ACM.

[24] A. Mandal, K. Kennedy, C. Koelbel, G. Marin, J. Mellor-Crummey, B. Liu, and L. Johnsson. Scheduling strategies for mapping application workflows onto the grid. In *High Performance Distributed Computing, 2005. HPDC-14. Proc. 14th IEEE Intl. Symp.*, pages 125–134, July 2005.

[25] R. Van den Bossche, K. Vanmechelen, and J. Broeckhove. Cost-optimal scheduling in hybrid iaas clouds for deadline constrained workloads. In *Cloud Computing (CLOUD), 2010 IEEE 3rd Intl. Conf. on*, pages 228–235, July 2010.

[26] Q. Tang, S.K.S. Gupta, and G. Varsamopoulos. Energy-efficient thermal-aware task scheduling for homogeneous high-performance computing data centers: A cyber-physical approach. *Parallel and Distributed Systems, IEEE Tr.*, 19(11):1458–1472, Nov 2008.

[27] K. Kc and K. Anyanwu. Scheduling hadoop jobs to meet deadlines. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE 2nd Intl. Conf. on*, pages 388–392, Nov 2010.

[28] Jun Wu, Xin Xu, Pengcheng Zhang, and Chunming Liu. A novel multi-agent reinforcement learning approach for job scheduling in grid computing. *Future Generation Computer Systems*, 27(5):430 – 439, 2011.

[29] Navendu Jain, Ishai Menache, Joseph (Seffi) Naor, and Jonathan Yaniv. A truthful mechanism for value-based scheduling in cloud computing. *Theor. Comp. Sys.*, 54(3):388–406, April 2014.

[30] Jia Yu, R. Buyya, and Chen Khong Tham. Cost-based scheduling of scientific workflow applications on utility grids. In *e-Science and Grid Computing, 2005. 1st Intl. Conf. on*, pages 8 pp.–147, July 2005.

[31] Dan Tsafrir, Yoav Etsion, and Dror G. Feitelson. Backfilling using system-generated predictions rather than user runtime estimates. *IEEE Tr. Parallel and Distrib Systems*, 18(6):789–803, 2007.

[32] S.K. Panda and P.K. Jana. A multi-objective task scheduling algorithm for heterogeneous multi-cloud environment. In *Electronic Design, Computer Networks Automated Verification (EDCAV), 2015 Intl. Conf. on*, pages 82–87, Jan 2015.

[33] J. Polo, D. Carrera, Y. Becerra, J. Torres, E. Ayguade, M. Steinder, and I. Whalley. Performance-driven task co-scheduling for mapreduce environments. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, pages 373–380, April 2010.

[34] Steven Y. Ko, Ramsés Morales, and Indranil Gupta. New worker-centric scheduling strategies for data-intensive grid applications. In Renato Cerqueira and Roy H. Campbell, editors, *Middleware 2007, ACM/IFIP/USENIX 8th Intl. Middleware Conference, Newport Beach, CA, USA, November 26-30, 2007, Proc.*, volume 4834 of *Lecture Notes in Computer Science*, pages 121–142. Springer, 2007.

[35] Staffs Keele. Guidelines for performing systematic literature reviews in software engineering. Technical report, Technical report, EBSE Tech. Rep. EBSE-2007-01, 2007.

[36] Matei Zaharia, Andy Konwinski, Anthony D. Joseph, Randy Katz, and Ion Stoica. Improving mapreduce performance in heterogeneous environments. In *Proc. of the 8th USENIX Conference on Operating Systems Design and Implementation*, OSDI'08, pages 29–42, Berkeley, CA, USA, 2008. USENIX Association.

[37] Anna Gorbenko and Vladimir Popov. Task-resource scheduling problem. *Intl. J. of Automation and Computing*, 9(4):429–441, 2012.

[38] Xin Liu, Chunming Qiao, Wei Wei, Xiang Yu, Ting Wang, Weisheng Hu, Wei Guo, and Min-You Wu. Task scheduling and lightpath establishment in optical grids. *Lightwave Technology, J. of*, 27(12):1796–1805, June 2009.

[39] Hongbo Liu, Ajith Abraham, and Aboul Ella Hassanien. Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm. *Future Generation Computer Systems*, 26(8):1336 – 1343, 2010.

[40] Zhangjun Wu, Zhiwei Ni, Lichuan Gu, and Xiao Liu. A revised discrete particle swarm optimization for cloud workflow scheduling. In *Computational Intelligence and Security (CIS), 2010 Intl. Conf. on*, pages 184–188, Dec 2010.

[41] Ruay-Shiung Chang, Jih-Sheng Chang, and Po-Sheng Lin. An ant algorithm for balanced job scheduling in grids. *Future Generation Computer Systems*, 25(1):20 – 27, 2009.

[42] Wei-Neng Chen and Jun Zhang. An ant colony optimization approach to a grid workflow scheduling problem with various qos requirements. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Tr.*, 39(1):29–43, Jan 2009.

[43] Lizhe Wang, Samee U. Khan, Dan Chen, Joanna Koodziej, Rajiv Ranjan, Cheng zhong Xu, and Albert Zomaya. Energy-aware parallel task scheduling in a cluster. *Future Generation Computer Systems*, 29(7):1661 – 1670, 2013. Including Special

sections: Cyber-enabled Distributed Computing for Ubiquitous. Cloud and Network Services & Cloud Computing and Scientific Applications - Big Data, Scalable Analytics, and Beyond.

[44] Lizhe Wang, G. von Laszewski, J. Dayal, and Fugang Wang. Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with dvfs. In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM Intl. Conf. on*, pages 368–377, May 2010.

[45] Fatma A. Omara and Mona M. Arafa. Genetic algorithms for task scheduling problem. *J. of Parallel and Distributed Computing*, 70(1):13 – 22, 2010.

[46] Yang Gao, Hongqiang Rong, and Joshua Zhexue Huang. Adaptive grid job scheduling with genetic algorithms. *Future Generation Computer Systems*, 21(1):151 – 161, 2005.

[47] Joanna Koodziej and Samee Ullah Khan. Multilevel hierarchic genetic-based scheduling of independent jobs in dynamic heterogeneous grid environment. *Information Sciences*, 214(0):1 – 19, 2012.

[48] Shaolei Ren, Yuxiong He, and Fei Xu. Provably-efficient job scheduling for energy and fairness in geographically distributed data centers. In *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd Intl. Conf. on*, pages 22–31, June 2012.

[49] Xiaomin Zhu, Chuan He, Kenli Li, and Xiao Qin. Adaptive energy-efficient scheduling for real-time tasks on dvs-enabled heterogeneous clusters. *J. of Parallel and Distributed Computing*, 72(6):751 – 763, 2012.

[50] S. Pandey, Linlin Wu, S.M. Guru, and R. Buyya. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE Intl. Conf.*, pages 400–407, April 2010.

[51] Joel Wolf, Deepak Rajan, Kirsten Hildrum, Rohit Khandekar, Vibhore Kumar, Sujay Parekh, Kun-Lung Wu, and Andrey balmin. Flex: A slot allocation scheduling optimizer for mapreduce workloads. In *Proc. of the ACM/IFIP/USENIX 11th Intl. Conf. on Middleware*, Middleware '10, pages 1–20, Berlin, Heidelberg, 2010. Springer-Verlag.

[52] Balaji Palanisamy, Aameek Singh, Ling Liu, and Bhushan Jain. Purlieus: Locality-aware resource allocation for mapreduce in a cloud. In *Proc. of 2011 Intl. Conf. for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 58:1–58:11, New York, NY, USA, 2011. ACM.

[53] Xiao Qin and Hong Jiang. A dynamic and reliability-driven scheduling algorithm for parallel real-time jobs executing on heterogeneous clusters. *J. of Parallel and Distributed Computing*, 65(8):885 – 900, 2005.

[54] Jordà Polo, Claris Castillo, David Carrera, Yolanda Becerra, Ian Whalley, Malgorzata Steinder, Jordi Torres, and Eduard Ayguadé. Resource-aware adaptive scheduling for mapreduce clusters. In *Proc. of the 12th ACM/IFIP/USENIX Intl. Conf. on Middleware*, Middleware'11, pages 187–207, Berlin, Heidelberg, 2011. Springer-Verlag.

[55] Xiangzhen Kong, Chuang Lin, Yixin Jiang, Wei Yan, and Xiaowen Chu. Efficient dynamic task scheduling in virtualized data centers with fuzzy prediction. *J. of Network and Computer Applications*, 34(4):1068 – 1077, 2011. Advanced Topics in Cloud Computing.

[56] Tridib Mukherjee, Ayan Banerjee, Georgios Varsamopoulos, Sandeep K.S. Gupta, and Sanjay Rungta. Spatio-temporal thermal-aware job scheduling to minimize energy consumption in virtualized heterogeneous data centers. *Computer Networks*, 53(17):2888 – 2904, 2009. Virtualized Data Centers.

[57] Chenhong Zhao, Shanshan Zhang, Qingfeng Liu, Jian Xie, and Jicheng Hu. Independent tasks scheduling based on genetic algorithm in cloud computing. In *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th Intl. Conf. on*, pages 1–4, Sept 2009.

[58] Abhishek Verma, Ludmila Cherkasova, and Roy H. Campbell. Aria: Automatic resource inference and allocation for mapreduce environments. In *Proc. of the 8th ACM Intl. Conf. on Autonomic Computing*, ICAC '11, pages 235–244, New York, NY, USA, 2011. ACM.

[59] Baris Aksanli, Jagannathan Venkatesh, Liuyi Zhang, and Tajana Rosing. Utilizing green energy prediction to schedule mixed batch and service jobs in data centers. *SIGOPS Oper. Syst. Rev.*, 45(3):53–57, January 2012.

[60] S. Selvarani and G.S. Sadhasivam. Improved cost-based algorithm for task scheduling in cloud computing. In *Computational Intelligence and Computing Research (ICCIC), 2010 IEEE Intl. Conf. on*, pages 1–5, Dec 2010.

[61] Jiahui Jin, Junzhou Luo, Aibo Song, Fang Dong, and Runqun Xiong. Bar: An efficient data locality driven task scheduling algorithm for cloud computing. In *Proc. of the 2011 11th IEEE/ACM Intl. Symp. Cluster, Cloud and Grid Computing*, CCGRID

'11, pages 295–304, Washington, DC, USA, 2011. IEEE Computer Society.

[62] Xiangping Bu, Jia Rao, and Cheng-zhong Xu. Interference and locality-aware task scheduling for mapreduce applications in virtual clusters. In *Proc. of the 22nd Intl. Symp. High-performance Parallel and Distributed Computing*, HPDC '13, pages 227–238, New York, NY, USA, 2013. ACM.

[63] Mohammad I. Daoud and Nawwaf Kharma. A high performance algorithm for static task scheduling in heterogeneous distributed computing systems. *J. of Parallel and Distributed Computing*, 68(4):399 – 409, 2008.

[64] Kento Aida and Henri Casanova. Scheduling mixed-parallel applications with advance reservations. *Cluster Computing*, 12(2):205–220, 2009.

[65] Vignesh T. Ravi, Michela Becchi, Wei Jiang, Gagan Agrawal, and Srimat Chakradhar. Scheduling concurrent applications on a cluster of cpugpu nodes. *Future Generation Computer Systems*, 29(8):2262 – 2271, 2013.

[66] Ziliang Zong, A. Manzanares, Xiaojun Ruan, and Xiao Qin. Ead and pebd: Two energy-aware duplication scheduling algorithms for parallel tasks on homogeneous clusters. *Computers, IEEE Tr.*, 60(3):360–374, March 2011.

[67] Saurabh Kumar Garg, Chee Shin Yeo, Arun Anandasivam, and Rajkumar Buyya. Environment-conscious scheduling of {HPC} applications on distributed cloud-oriented data centers. *J. of Parallel and Distributed Computing*, 71(6):732 – 749, 2011. Special Issue on Cloud Computing.

[68] Kyong Hoon Kim, R. Buyya, and Jong Kim. Power aware scheduling of bag-of-tasks applications with deadline constraints on dvs-enabled clusters. In *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE Intl. Symp.*, pages 541–548, May 2007.

[69] Suchang Guo, Hong-Zhong Huang, Zhonglai Wang, and Min Xie. Grid service reliability modeling and optimal task scheduling considering fault recovery. *Reliability, IEEE Tr.*, 60(1):263–274, March 2011.

[70] Elizeu Santos-Neto, Walfredo Cirne, Francisco Brasileiro, and Aliandro Lima. Exploiting replication and data reuse to efficiently schedule data-intensive applications on grids. In DrorG. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing*, volume 3277 of *Lecture Notes in Computer Science*, pages 210–232. Springer Berlin Heidelberg, 2005.

[71] J. Octavio Gutierrez-Garcia and Kwang Mong Sim. A family of heuristics for agent-based elastic cloud bag-of-tasks concurrent scheduling. *Future Generation Computer Systems*, 29(7):1682 – 1699, 2013. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services & Cloud Computing and Scientific Applications Big Data, Scalable Analytics, and Beyond.

[72] Thamarai Selvi Somasundaram and Kannan Govindarajan. Cloudrb: A framework for scheduling and managing high-performance computing (hpc) applications in science cloud. *Future Generation Computer Systems*, 34(0):47 – 65, 2014. Special Section: Distributed Solutions for Ubiquitous Computing and Ambient Intelligence.

[73] K. Etminani and M. Naghibzadeh. A min-min max-min selective algorihtm for grid task scheduling. In *Internet, 2007. ICI 2007. 3rd IEEE/IFIP Intl. Conf. in Central Asia on*, pages 1–7, Sept 2007.

[74] Jia Yu and R. Buyya. A budget constrained scheduling of workflow applications on utility grids using genetic algorithms. In *Workflows in Support of Large-Scale Science, 2006. WORKS '06. Wkshp.*, pages 1–10, June 2006.

[75] S. Abrishami, M. Naghibzadeh, and D.H.J. Epema. Cost-driven scheduling of grid workflows using partial critical paths. *Parallel and Distributed Systems, IEEE Tr.*, 23(8):1400–1414, Aug 2012.

[76] Fan Zhang, Junwei Cao, Keqin Li, Samee U. Khan, and Kai Hwang. Multi-objective scheduling of many tasks in cloud platforms. *Future Generation Computer Systems*, 37(0):309 – 320, 2014. Special Section: Innovative Methods and Algorithms for Advanced Data-Intensive Computing Special Section: Semantics, Intelligent processing and services for big data Special Section: Advances in Data-Intensive Modelling and Simulation Special Section: Hybrid Intelligence for Growing Internet and its Applications.

[77] M. Mezmaz, N. Melab, Y. Kessaci, Y.C. Lee, E.-G. Talbi, A.Y. Zomaya, and D. Tuyttens. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. *J. of Parallel and Distributed Computing*, 71(11):1497 – 1508, 2011.

[78] Luiz Fernando Bittencourt and Edmundo Roberto Mauro Madeira. Hcoc: a cost optimization algorithm for workflow scheduling in hybrid clouds. *J. of Internet Services and Applications*, 2(3):207–227, 2011.

[79] M.A. Rodriguez and R. Buyya. Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. *Cloud Computing, IEEE Tr.*, 2(2):222–235, April 2014.

[80] T.A.L. Genez, L.F. Bittencourt, and E.R.M. Madeira. Workflow scheduling for saas / paas cloud providers considering two sla levels. In *Network Operations and Management Symposium (NOMS), 2012 IEEE*, pages 906–912, April 2012.

[81] Palden Lama and Xiaobo Zhou. Aroma: Automated resource allocation and configuration of mapreduce environment in the cloud. In *Proc. of the 9th Intl. Conf. on Autonomic Computing*, ICAC '12, pages 63–72, New York, NY, USA, 2012. ACM.

[82] Srikumar Venugopal, Rajkumar Buyya, and Kotagiri Ramamohanarao. A taxonomy of data grids for distributed data sharing, management, and processing. *ACM Comput. Surv.*, 38(1), June 2006.

[83] Joanna Koodziej and Samee Ullah Khan. Data scheduling in data grids and data centers: A short taxonomy of problems and intelligent resolution techniques. In Ngoc-Thanh Nguyen, Joanna Koodziej, Tadeusz Burczyski, and Marenglen Biba, editors, *Tr. Computational Collective Intelligence X*, volume 7776 of *Lecture Notes in Computer Science*, pages 103–119. Springer Berlin Heidelberg, 2013.

[84] Jia Yu, Rajkumar Buyya, and Kotagiri Ramamohanarao. Workflow scheduling algorithms for grid computing. In Fatos Xhafa and Ajith Abraham, editors, *Metaheuristics for Scheduling in Distributed Computing Environments*, volume 146 of *Studies in Computational Intelligence*, pages 173–214. Springer Berlin Heidelberg, 2008.

[85] Nikolay Grozev and Rajkumar Buyya. Inter-cloud architectures and application brokering: taxonomy and survey. *Software: Practice and Experience*, 44(3):369–390, 2014.

[86] Chee Shin Yeo and Rajkumar Buyya. A taxonomy of market-based resource management systems for utility-driven cluster computing. *Software: Practice and Experience*, 36(13):1381–1419, November 2006.

[87] Jiannong Cao, Alvin T. Chan, Yudong Sun, Sajal K. Das, and Minyi Guo. A taxonomy of application scheduling tools for high performance cluster computing. *Cluster Computing*, 9(3):355–371, July 2006.

[88] Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn. Parallel job scheduling - A status report. In Dror G. Feitelson, Larry Rudolph, and Uwe Schwiegelshohn, editors, *Job Scheduling Strategies for Parallel Processing, 10th Intl. Wkshp., JSSPP 2004, New York, NY, USA, June 13, 2004, Revised Selected Papers*, volume 3277 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2004.

[89] Fatos Xhafa and Ajith Abraham. Computational models and heuristic methods for grid scheduling problems. *Future Generation Computer Systems*, 26(4):608 – 621, 2010.

[90] Hameed Hussain, Saif Ur Rehman Malik, Abdul Hameed, Samee Ullah Khan, Gage Bickler, Nasro Min-Allah, Muhammad Bilal Qureshi, Limin Zhang, Yongji Wang, Nasir Ghani, Joanna Kolodziej, Albert Y. Zomaya, Cheng-Zhong Xu, Pavan Balaji, Abhinav Vishnu, Frédéric Pinel, Johnatan E. Pecero, Dzmitry Kliazovich, Pascal Bouvry, Hongxiang Li, Lizhe Wang, Dan Chen, and Ammar Rayes. A survey on resource allocation in high performance distributed computing systems. *Parallel Computing*, 39(11):709–736, 2013.

[91] Elina Pacini, Cristian Mateos, and Carlos Garca Garino. Distributed job scheduling based on swarm intelligence: A survey. *Computers & Electrical Engineering*, 40(1):252 – 269, 2014. 40th-year commemorative issue.

[92] Li Liu, Miao Zhang, Yuqing Lin, and Liangjuan Qin. A survey on workflow management and scheduling in cloud computing. In *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM Intl. Symp.*, pages 837–846, May 2014.

[93] Zhangjun Wu, Xiao Liu, Zhiwei Ni, Dong Yuan, and Yun Yang. A market-oriented hierarchical scheduling strategy in cloud workflow systems. *The J. of Supercomputing*, 63(1):256–293, 2013.

[94] Saeid Abrishami, Mahmoud Naghibzadeh, and Dick H.J. Epema. Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. *Future Generation Computer Systems*, 29(1):158 – 169, 2013. Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures.

[95] David A. Lifka. The anl/ibm sp scheduling system. In *Proc. of the Wkshp. Job Scheduling Strategies for Parallel Processing*, IPPS '95, pages 295–303, London, UK, UK, 1995. Springer-Verlag.

[96] Ioannis A. Moschakis and Helen D. Karatza. Evaluation of gang scheduling performance and cost in a cloud computing system. *J. Supercomput.*, 59(2):975–992, February 2012.

[97] Meng Xu, Lizhen Cui, Haiyang Wang, and Yanbing Bi. A multiple qos constrained scheduling

strategy of multiple workflows for cloud computing. In *Parallel and Distributed Processing with Applications, 2009 IEEE Intl. Symp.*, pages 629–634, Aug 2009.

[98] S. Song, Kai Hwang, and Yu-Kwong Kwok. Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling. *Computers, IEEE Tr.*, 55(6):703–719, June 2006.

[99] Ming Tang, Bu-Sung Lee, Xueyan Tang, and Chai-Kiat Yeo. The impact of data replication on job scheduling performance in the data grid. *Future Generation Computer Systems*, 22(3):254 – 268, 2006.

[100] Jin Xu, A.Y.S. Lam, and V.O.K. Li. Chemical reaction optimization for task scheduling in grid computing. *Parallel and Distributed Systems, IEEE Tr.*, 22(10):1624–1631, Oct 2011.

[101] Yun-Han Lee, Seiven Leu, and Ruay-Shiung Chang. Improving job scheduling algorithms in a grid environment. *Future Generation Computer Systems*, 27(8):991 – 998, 2011.

[102] Quan Chen, Daqiang Zhang, Minyi Guo, Qianni Deng, and Song Guo. Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment. In *Computer and Information Technology (CIT), 2010 IEEE 10th Intl. Conf. on*, pages 2736–2743, June 2010.

[103] Wei Wang, Guosun Zeng, Daizhong Tang, and Jing Yao. Cloud-dls: Dynamic trusted scheduling for cloud computing. *Expert Systems with Applications*, 39(3):2321 – 2329, 2012.

[104] Javid Taheri, Young Choon Lee, Albert Y. Zomaya, and Howard Jay Siegel. A bee colony based optimization approach for simultaneous job scheduling and data replication in grid environments. *Computers & Operations Research*, 40(6):1564 – 1578, 2013. Emergent Nature Inspired Algorithms for Multi-Objective Optimization.

[105] Hyunseok Chang, M. Kodialam, R.R. Kompella, T.V. Lakshman, Myungjin Lee, and S. Mukherjee. Scheduling in mapreduce-like systems for fast completion time. In *INFOCOM, 2011 Proc. IEEE*, pages 3074–3082, April 2011.

[106] Saurabh Kumar Garg, Rajkumar Buyya, and H. J. Siegel. Scheduling parallel applications on utility grids: Time and cost trade-off management. In *Proc. of the Thirty-2nd Australasian Conference on Computer Science - Volume 91*, ACSC '09, pages 151–160, Darlinghurst, Australia, Australia, 2009. Australian Computer Society, Inc.

[107] M. Hammoud and M.F. Sakr. Locality-aware reduce task scheduling for mapreduce. In *Cloud Computing Technology and Science (CloudCom), 2011 IEEE 3rd Intl. Conf. on*, pages 570–576, Nov 2011.

[108] Ruay-Shiung Chang, Jih-Sheng Chang, and Shin-Yi Lin. Job scheduling and data replication on data grids. *Future Generation Computer Systems*, 23(7):846 – 860, 2007.

[109] Katia Leal, Eduardo Huedo, and Ignacio M. Llorente. A decentralized model for scheduling independent tasks in federated grids. *Future Generation Computer Systems*, 25(8):840 – 852, 2009.

[110] Adán Hirales-Carbajal, Andrei Tchernykh, Ramin Yahyapour, José Luis González-García, Thomas Röblitz, and Juan Manuel Ramírez-Alcaraz. Multiple workflow scheduling strategies with user run time estimates on a grid. *J. Grid Comput.*, 10(2):325–346, June 2012.

[111] Chuliang Weng and Xinda Lu. Heuristic scheduling for bag-of-tasks applications in combination with qos in the computational grid. *Future Generation Computer Systems*, 21(2):271 – 280, 2005. Advanced Grid Technologies.

[112] T. Hagras and J. Janeek. A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems. *Parallel Computing*, 31(7):653 – 670, 2005. Heterogeneous Computing.

[113] A.J. Page and T.J. Naughton. Dynamic task scheduling using genetic algorithms for heterogeneous distributed computing. In *Parallel and Distributed Processing Symposium, 2005. Proc. 19th IEEE Intl.*, pages 189a–189a, April 2005.

[114] Radu Prodan and Thomas Fahringer. Dynamic scheduling of scientific workflow applications on the grid: A case study. In *Proc. of the 2005 ACM Symp. Applied Computing*, SAC '05, pages 687–694, New York, NY, USA, 2005. ACM.

[115] Fatos Xhafa, Enrique Alba, Bernab Dorronsoro, and Bernat Duran. Efficient batch job scheduling in grids using cellular memetic algorithms. *J. of Mathematical Modelling and Algorithms*, 7(2):217–236, 2008.

[116] Niels Fallenbeck, Hans-Joachim Picht, Matthew Smith, and Bernd Freisleben. Xen and the art of cluster scheduling. In *Proc. of the 2nd Intl. Wkshp. Virtualization Technology in Distributed Computing*, VTDC '06, pages 4–, Washington, DC, USA, 2006. IEEE Computer Society.

[117] M. Rahman, S. Venugopal, and R. Buyya. A dynamic critical path algorithm for scheduling scientific workflow applications on global grids. In *e-Science and Grid Computing, IEEE Intl. Conf. on*, pages 35–42, Dec 2007.

[118] O. Sinnen, L.A. Sousa, and F.E. Sandnes. Toward a realistic task scheduling model. *Parallel and Distributed Systems, IEEE Tr.*, 17(3):263–275, March 2006.

[119] Javad Akbari Torkestani. A new approach to the job scheduling problem in computational grids. *Cluster Computing*, 15(3):201–210, 2012.

[120] Young Choon Lee and A.Y. Zomaya. Practical scheduling of bag-of-tasks applications on grids with dynamic resilience. *Computers, IEEE Tr.*, 56(6):815–825, June 2007.

[121] A. Verma, L. Cherkasova, and R.H. Campbell. Two sides of a coin: Optimizing the schedule of mapreduce jobs to minimize their makespan and improve cluster performance. In *Modeling, Analysis Simulation of Computer and Telecommunication Systems (MASCOTS), 2012 IEEE 20th Intl. Symp.*, pages 11–18, Aug 2012.

[122] Stylianos Zikos and Helen D. Karatza. Performance and energy aware cluster-level scheduling of compute-intensive jobs with unknown service times. *Simulation Modelling Practice and Theory*, 19(1):239 – 250, 2011. Modeling and Performance Analysis of Networking and Collaborative Systems.

[123] Jinn-Tsong Tsai, Jia-Cen Fang, and Jyh-Horng Chou. Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. *Computers & Operations Research*, 40(12):3045 – 3055, 2013.

[124] JuanManuel Ramrez-Alcaraz, Andrei Tchernykh, Ramin Yahyapour, Uwe Schwiegelshohn, Ariel Quezada-Pina, JosLuis Gonzlez-Garca, and Adn Hirales-Carbajal. Job allocation strategies with user run time estimates for online scheduling in hierarchical grids. *J. of Grid Computing*, 9(1):95–116, 2011.

[125] Kento Aida and Henri Casanova. Scheduling mixed-parallel applications with advance reservations. *Cluster Computing*, 12(2):205–220, 2009.

[126] Xuan Lin, Y. Lu, J. Deogun, and S. Goddard. Real-time divisible load scheduling for cluster computing. In *Real Time and Embedded Technology and Applications Symposium, 2007. RTAS '07. 13th IEEE*, pages 303–314, April 2007.

[127] Xin Liu, Chunming Qiao, Wei Wei, Xiang Yu, Ting Wang, Weisheng Hu, Wei Guo, and Min-You Wu. Task scheduling and lightpath establishment in optical grids. *Lightwave Technology, J. of*, 27(12):1796–1805, June 2009.

A The Review Protocol

Our survey follows the general guidelines for performing systematic literature reviews in software engineering proposed by the Software Engineering Group at the University of Keele [35]. The process includes the identification of the research questions and a review protocol. The following research questions served as the basis for the systematic literature review.

1. *RQ1*: How to define a scheduling problem in the context of parallel job scheduling in distributed platforms?

2. *RQ2*: How to classify scheduling solutions in the context of parallel job scheduling in distributed platforms?

3. *RQ3*: What are the most popular scheduling problems being investigated in the past ten years in the context of parallel job scheduling in distributed platforms?

4. *RQ4*: What are the most popular classes of scheduling solutions proposed in the past ten years in the context of parallel job scheduling in distributed platforms?

A.1 Strategy

Our search *strategy* looked for the following terms in the paper title: (scheduling $\vee$ schedule $\vee$ allocation $\vee$ scheduler) *and* (cloud $\vee$ grid $\vee$ cluster $\vee$ "data center") *and* (job $\vee$ application $\vee$ DAG $\vee$ MapReduce $\vee$ hadoop $\vee$ task $\vee$ workflow). The repositories considered for this part of the search were the ACM digital library, SpringerLink, IEEExplorer, and ScienceDirect.

We also collected a second set of papers using the following search strategy: highly cited papers (i.e., more than 8 citations per year since publication), obtained from Google scholar, not found during the previous search, and that satisfy the following relaxed search terms: (scheduling $\vee$ schedule $\vee$ allocation $\vee$ scheduler) *and* (cloud $\vee$ grid $\vee$ cluster $\vee$ "data center" $\vee$ job $\vee$ application $\vee$ DAG $\vee$ MapReduce $\vee$ hadoop $\vee$ task $\vee$ workflow). We call these papers the set of *GS papers*.

Papers to be collected must satisfy the one of the above criteria and must have been published from January 1^{st}, 2005 to May, 1^{st}, 2015.

A.2 Exclusion criteria

Some papers that match the search strategy may be excluded. We consider two different exclusion criteria: exclusion by keywords in the title and exclusion by content.

A.2.1 Exclusion by key words in the title

Papers whose title contain one of the following terms were excluded: comparison, smart grid, job-shop, manufacturing, simulation, mobile, and architecture. These papers are not related to the subject of interest or present comparisons or evaluations of solutions presented in other papers.

A.2.2 Exclusion by content

The following exclusion criteria were applied after the papers were read.

- If the scheduling solution was published in more than one venue, we used the most complete version of the work;
- Papers describing related surveys and/or taxonomies were excluded[13];
- Papers that do not investigate scheduling of parallel jobs in distributed environments were excluded for not matching our review interests.

A.3 Study selection process

Five steps were used to obtain the information needed to answer our research questions.

A.3.1 Step 1: Paper collection

Perform a manual search using the search strategy above. The following items were collected for each paper: year of publication, authors, title, source of publication, link to the paper, and number of citations (collected manually from Google Scholar).

A.3.2 Step 2: Paper exclusion by keywords

Exclude the papers that satisfy at least one of the exclusion criteria by keywords in the title as described in subsection A.2.1.

A.3.3 Step 3: Computation of citations per year

Compute the number of citations per year for each paper, CPY_{paper}, by dividing the total number of citations, NC_{paper}, the paper received since its publication up to the day the data was collected by the number of years since its publication:

$$CPY_{paper} = \frac{NC_{paper}}{2015.33 - Y_{paper}}. \quad (2)$$

The number of years since publication in the denominator of Equation 2 is computed as the difference between 2015.33 (to represent the data collection date of May 1, 2015) and the year the paper was published.

A.3.4 Step 4: Impact Analysis

Perform a statistical analysis of the remaining set of papers to identify the main resource categories considered for scheduling and assess the impact of these papers.

A.3.5 Step 5: Classification of 100 problems

Sort all the papers in descending order of number of citations per year. Read the most cited papers in order to classify the scheduling problems and solutions according to our taxonomy. Reading the papers may lead to the exclusion of some of them due to the exclusion criteria by content (Subsection A.2.2). Continue reading until 100 problems are classified. The number of papers included in this step cannot be anticipated because it depends on the number of problems investigated in each paper and the number of papers excluded by content.

It is important to mark features that are not classified with confidence in each paper in order to identify features of scheduling problems and solutions that are not properly defined by the authors (see Section 6.2). Extract the following additional information from each paper classified: the source (i.e., the conference or journal) and the name of the first author.

These five steps produce three deliverables: (i) the impact analysis of the papers collected (excluding some papers by applying exclusion criteria presented in A.2.1); (ii) 100 scheduling problems and solutions classified according to our taxonomy; and (iii) information to perform an analysis of the features of problems and solutions that are currently identified with less confidence.

A.4 Data extraction strategy

The data extracted from each paper during the paper collection (step 1) was used for an impact analysis. Some questions of interest are:

- How many papers of interest per year were published over these 10 years?
- Is the number of papers published in this area stable over the years?
- What is the fraction of papers explicitly related to clusters, grids, clouds, and data centers?

[13] These papers were considered as related work.

- Grouping data by year, what is the fraction of papers explicitly related to clusters, grids, clouds, and data centers?
  - Do these relations change with time? Are there significant differences related to the main resource platform considered over the years?
- How does the number of citations per year change over time?
- What is the fraction of papers that explicitly address MapReduce applications or Hadoop?
- What is the average popularity of papers in this area?
- Is the popularity of the papers in this area stable over the years?

After the 100 problems are classified applying the taxonomy, a *clustering analysis* must be carried out to determine what are the most common scheduling problems being investigated in the past 10 years in the context of job scheduling in distributed computing systems.

A.5 Dissemination strategy

The results of such a study must be published in a reputable journal in order to have high impact.

B List of top-100 problems

1. HFS [21];
2. QFP [22];
3. Swarm-Opt [50];
4. GMCE/GMP [67];
5. ACO/GA/PSO [93];
6. ARIA [58];
7. XInt [26];
8. GreenHadoop [23];
9. DMDP [30];
10. IC-PCP/IC-PDPD2 [94];
11. ACS [42];
12. BIP [25];
13. HybridGA [77];
14. PB [31, 95];
15. SS-EDF/PShare [68];
16. DCLS/AMMS-(EL) [20];
17. PALS/PATC [43];
18. BACO [41];
19. Min/Max [33];
20. Min/Max/Suff [67];
21. Purlieus [52];
22. CPGA/TDGA [45];
23. BEF [44];
24. EPTS [44];
25. AFCFS/LJFS [96];
26. FDPSO [39];
27. PCP [75];
28. HCOC [78];
29. HAS [24];
30. TTS [27];
31. ALS-GA [46];
32. MQMW [97];
33. BFS [29];
34. DT-STGA [98];
35. STT [99];
36. FCFS-Backfill-XInt [56];
37. SCINT [56];
38. GA [74];
39. CRO [100];
40. SAT/PSOE [37];
41. FLEX [51];
42. PSO [79];
43. VOO [76];
44. Pred [59];
45. Selective [73];
46. HLBA [101];
47. RAS [54];
48. SAMR [102];
49. Cloud-DLS [103];
50. GreFar [48];

51. LDCP [63];
52. MOTS [32];
53. JDS-BC [104];
54. OSLS [28];
55. ONA [105];
56. MinCTT/MaxCTT [106];
57. LARTS [107];
58. HCS [108];
59. BMT/BMEMT [80];
60. GAS [57];
61. DO-AS [109];
62. StorageAffinity [70];
63. BAR [61];
64. RDPSO [40];
65. MWGS [110];
66. EAD/PEBD [66];
67. QSufferage [111];
68. HCPFD [112];
69. PN [113];
70. IABC [60];
71. ILA [62];
72. DS [114];
73. cMA+LMCTS/LTH [115];
74. XGE [116];
75. DCP-G [117];
76. ACO [69];
77. RSA/RSC/ASJ [65];
78. FMS [65];
79. CLOUDRB [72];
80. LS/invo-cont [118];
81. LAJS [119];
82. SIL [120];
83. MDQ [120];
84. SALAF [55];
85. HGS-Sched [47];
86. (Ltos,*)/(Stol,*) [71];
87. (U,*) [71];
88. AEES [49];
89. AROMA [81];
90. BalancedPools [121];
91. SQEE/SQHP/PBP-SQ [122];
92. DRCD [53];
93. IDEA [123];
94. MPL/LBal_S [124];
95. RESSCHED [125];
96. RESSCHEDDL [125];
97. DLT/EDF [126];
98. LS_APF/LS_P [127];
99. OPT [127];
100. LS/LS_CSP [127];
101. LS/LS_CSP-OPT [127];